JOURNAL OF COMPUTATIONAL PHYSICS **141,** 112–152 (1998)
ARTICLE NO. CP985906

Reconstructing Volume Tracking[1]

William J. Rider * and Douglas B. Kothe†

**Applied Theoretical and Computational Physics Division, Hydrodynamic Methods Group (X-HM),*
†Materials Science and Technology Division, Structure/Property Relations Group (MST-8),
Los Alamos National Laboratory, Los Alamos, New Mexico, 87545
E-mail: wjr@lanl.gov, dbk@lanl.gov

Received May 14, 1997; revised December 15, 1997

A new algorithm for the volume tracking of interfaces in two dimensions is presented. The algorithm is based upon a well-defined, second-order geometric solution of a volume evolution equation. The method utilizes local discrete material volume and velocity data to track interfaces of arbitrarily complex topology. A linearity-preserving, piecewise linear interface geometry approximation ensures that solutions generated retain second-order spatial accuracy. Second-order temporal accuracy is achieved by virtue of a multidimensional unsplit time integration scheme. We detail our geometrically based solution method, in which material volume fluxes are computed systematically with a set of simple geometric tasks. We then interrogate the method by testing its ability to track interfaces through large, controlled topology changes, whereby an initially simple interface configuration is subjected to vortical flows. Numerical results for these strenuous test problems provide evidence for the algorithm's improved solution quality and accuracy.

Key Words: volume of fluid; volume tracking; interface tracking.

1. INTRODUCTION

Volume tracking methods have enjoyed widespread use and success since the mid-1970s, yet they possess solution algorithms that are too often perceived as being heuristic and without mathematical formalism. Part of this misperception lies in the difficulty of applying standard hyperbolic PDE numerical analysis tools, which assume algebraic formulations, to a method that is largely geometric in nature (hence, the more appropriate term *volume tracking*). To some extent the lack of formalism in volume tracking methods, manifested as an obscure underlying methodology, has impeded progress in evolutionary algorithmic improvements.

[1] This work performed under the auspices of the U.S. Department of Energy by Los Alamos National Laboratory under Contract W-7405-ENG-36.

0021-9991/98 $25.00

In this paper we clarify the methodology underlying modern volume tracking methods, and, in the process, present a new two-dimensional algorithm that is constructed from a geometric solution of a volume evolution equation. The solutions are second order in space through the use of a linearity-preserving piecewise-linear interface geometry approximation. Second-order temporal accuracy is realized with a novel multidimensional unsplit time integration scheme. We describe our solution method whereby volume fluxes are computed with a set of straightforward geometric tasks.

Basic features of volume tracking methods. It is first instructive to review the common features of most volume tracking methods. To begin, fluid volumes are initialized in each computational cell from a specified interface geometry. This task requires computing fluid volumes in each cell containing the interface (hereafter referred to as *mixed* cells). Exact interface information is then discarded in favor of the discrete volume data. The volume data is traditionally retained as volume fractions (denoted as f hereafter), whereby mixed cells will have a volume fraction f between zero and one, and cells without interfaces (*pure* cells) will have a volume fraction f equal to zero or unity. Since a unique interface configuration does not exist once the exact interface location is replaced with discrete volume data, detailed interface information cannot be extracted until an interface is *reconstructed*. The principal reconstruction constraint is local volume conservation, i.e., the reconstructed interface must truncate cells with a volume equal to the discrete fluid volumes.

Interfaces are "tracked" in volume tracking methods by evolving fluid volumes forward in time with solutions of an advection equation. At any time in the solution, exact interface locations are not known; i.e., a given distribution of volume data does not guarantee a unique interface. Interface geometry must be inferred, based on local volume data and the assumptions of the particular algorithm, before interfaces can be reconstructed. The reconstructed interface is then used to compute the volume fluxes necessary to integrate the volume evolution equations. Typical implementations of these algorithms are one-dimensional, with multidimensionality obtained through operator splitting [1].

We begin in Section 2 with a historical perspective of volume tracking methods. Next, in Section 3, governing fluid volume equations are derived from basic principles. This derivation is straightforward for flows that are incompressible, but considerably more involved for flows with compressibility. In Section 4 our solution method is detailed with the help of a set of concise geometric operations. (A flow chart of the principal algorithmic steps discussed in Section 4 is shown in Fig. 1.) Finally, in Section 5, we interrogate our method by computing gross interface topology changes, whereby an initially simple interface configuration is subjected to flows with appreciable vorticity (see also Appendix B). An example of the performance of our volume-tracking method on a simpler test (one without topology changes) is shown in Fig. 2, where a slotted-cylinder [2] is shown before and after one revolution.

2. HISTORICAL PERSPECTIVE

Table 1 summarizes the salient features of notable volume tracking methods published since 1974. Listed for each method are important aspects of the interface reconstruction and volume advection algorithms. Identifiable reconstruction features include the assumed interface geometry, which tends to be either piecewise constant, piecewise constant/"stair-stepped," or piecewise linear; and the method used for computing the interface normal, which

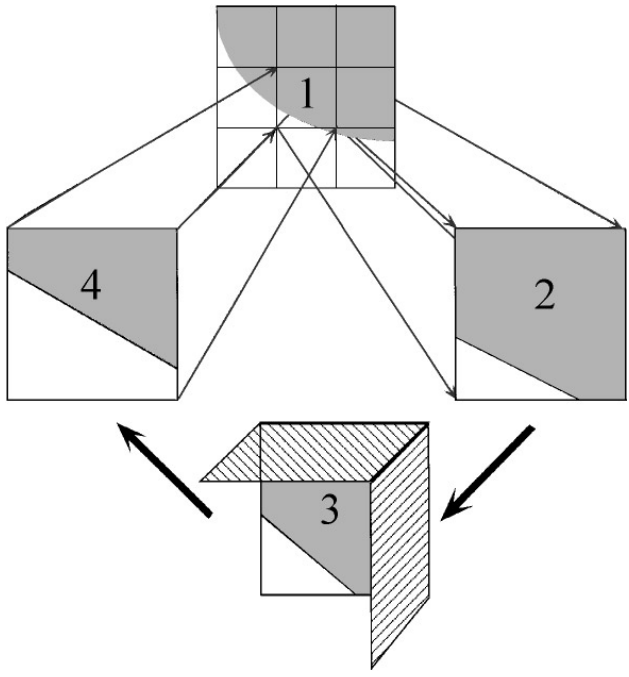

FIG. 1. Flow chart of the four basic steps comprising our volume-tracking method: (1) discrete material volume data is provided on the computational domain; (2) a piecewise linear interface is reconstructed (Section 4.2); (3) material volume fluxes are computed as truncation volumes (here shown for the unsplit integration discussed in Section 4.3); and (4) the volumes are integrated to a new time level (Section 4.4). Each step utilizes the geometric toolbox described in Section 4.1.

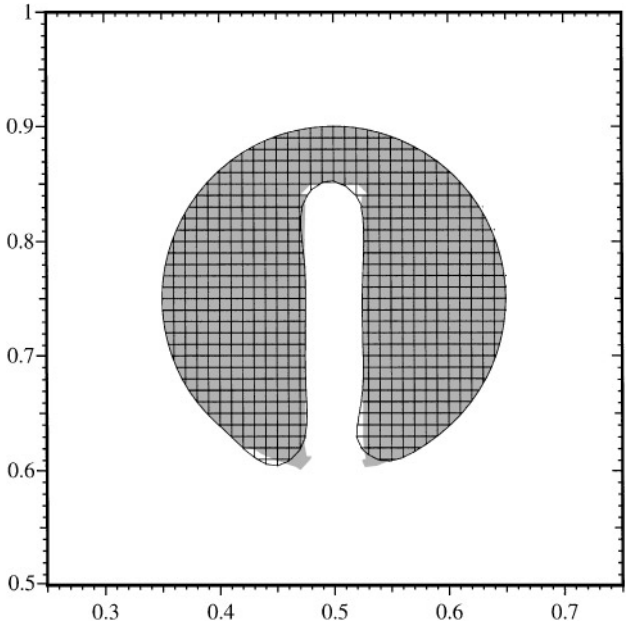

FIG. 2. Performance of our method on Zalesak's slotted-cylinder test problem [2] using a 100×100 grid. Shaded portions represent a reconstruction of the initial conditions, whereas the line segments delineate the slotted cylinder after one revolution. The shaded region boundary and line segments virtually overlay, except for high curvature regions (i.e., the sharp corners), where smoothing has occurred. The corner errors, however, are quite small in comparison to more traditional methods that cannot maintain a sharp interface.

TABLE 1
Reconstructed Interface Geometry and Time Integration Method Used in a Variety of Published Volume Tracking Algorithms

Author(s)	Reconstructed interface geometry	Time integration
DeBar [3]	Piecewise linear, operator split	Operator split
Noh and Woodward [4]	Piecewise constant, operator split	Operator split
Hirt and Nichols [5]	Piecewise constant, stair-stepped, multidimensional	Operator split
Chorin [6]	Piecewise constant, stair-stepped, multidimensional	Operator split
Barr and Ashurst [7]	Piecewise constant, stair-stepped, multidimensional	Operator split
Ashgriz and Poo [8]	Piecewise linear, operator split	Operator split
Youngs [9]	Piecewise linear, multidimensional	Operator split
Pilliod and Puckett [10, 11]	Piecewise linear, multidimensional	Multidimensional

is either one-dimensional (operator split) or multidimensional. Similarly, time integration of the volume advection equation can be constructed in an operator split or multidimensional fashion. Below we summarize briefly the chronology and impact of these developments.

2.1. *The Genesis of Volume Tracking*

Within a short period of time in the early 1970s, the first three volume tracking methods were introduced: DeBar's method [3], Hirt and Hichols' VOF (for volume-of-fluid) method [12, 13], and Noh and Woodward's SLIC (for simple line interface calculation) method [4]. Each of these methods chose a different reconstructed interface geometry: the DeBar algorithm used a piecewise linear approximation, the VOF method used a piecewise constant/"stair-stepped" approximation, and the SLIC algorithm invoked a piecewise constant approximation. As shown in Table 1, most volume tracking algorithms published to date fall into one of these three interface reconstruction categories; piecewise constant, piecewise constant/stair-stepped, or piecewise linear. DeBar's piecewise linear choice for the reconstructed interface geometry is still generally preferred in modern volume tracking algorithms.

The SLIC method approximates interfaces as piecewise constant, where interfaces within each cell are assumed to be lines (or planes in three dimensions) aligned with one of the logical mesh coordinates. This choice of a simpler interface geometry (relative to Debar's piecewise linear choice) appears to have been made to facilitate treatment of multiple (>2) materials within a given mixed cell. In piecewise constant/stair-stepped methods such as VOF, interfaces are also forced to align with mesh coordinates, but are additionally allowed to "stair-step" (align with more than one mesh coordinate) within each cell, depending upon the local distribution of discrete volume data.

Interface normals are computed in DeBar's method with one-dimensional volume fraction differences, i.e., by considering only those cells sharing a face across which volume fluxes are to be estimated. In this sense the reconstruction can be considered "operator split" since the interface normal follows from one-dimensional differences based upon the current advection sweep direction. A method similar to DeBar's method is described in the more recent work of Norman and Winkler [14], where it is attributed to earlier work of LeBlanc. The SLIC method, as in DeBar's method, estimates interface normals with operator split

differences.[1] Modern SLIC implementations have improved slightly via use of multidimensional operators (3×3 stencil in two dimensions) for the normal and center-of-mass coordinates to aid in placing the interface within the cell [15]. The VOF algorithm also uses a multidimensional operator in determining interface orientation. This information helps position the reconstructed stair-stepped interface within each cell.

Another pioneering development is attributed to James LeBlanc of Lawrence Livermore National Laboratory. LeBlanc's method, as described by Bowers and Wilson [16, 17], has an algebraic (rather than geometric) basis. LeBlanc's scheme for the transport of volumes to and from mixed cells (the so-called "mixed-to-mixed" cell transport), however, also has a geometric interpretation. Close scrutiny of this mode of transport reveals that LeBlanc's "area" factors are nearly identical to those used in a subsequent piecewise linear scheme devised by Youngs (discussed later) [9].

Many early volume tracking methods were devised and formulated algebraically (i.e., using combinations of upwind and downwind fluxes) rather than geometrically. In fact, some low-order methods (e.g., SLIC or VOF) can be derived a number of ways: algebraically, geometrically, or heuristically. The chosen derivation for these cases is largely a matter of taste. For piecewise linear methods, however, a geometric framework is preferable because concise algebraic descriptions can be difficult, especially in three dimensions. To facilitate comparisons with our piecewise linear method, therefore, we will interpret volume tracking methods geometrically where possible, while keeping in mind this interpretation might be contrary to the original authors' philosophy. For a current reference on algebraic approaches to volume tracking methods, see the recent paper by Rudman [18].

2.2. *Piecewise Constant Volume Tracking Methods*

Many piecewise constant volume tracking algorithms have been published subsequent to the VOF and SLIC algorithms, e.g., see Chorin [6] and Barr and Ashurst [7]. As seen in Table 1, newer versions of these methods offered improvements such as a multidimensional reconstruction algorithm, but operator-split time integration techniques have still been relied upon. In general, these methods have been evolutionary, but still retained the simplistic piecewise constant geometry assumption.

A notable feature of the VOF method is that its volume fluxes can be formulated algebraically, i.e., without needing an exact reconstructed interface position. The volume fluxes can be expressed as a weighted sum of upwind and downwind contributions, depending upon the orientation of the interface relative to the local flow direction. If the reconstructed interface is parallel to the flow, an upwind flux is used; otherwise a portion of downwind flux is added to counteract numerical diffusion brought about by the piecewise constant upwinding. This approach, which falls into the general family of flux-corrected transport methods [2, 19] (simplifying its analysis), was the underlying theme behind the design of the VOF method. This flux-limiting methodology has also been used recently to define modern variants, e.g., see [18, 20].

A feature characteristic of piecewise constant volume tracking methods (with or without stair-stepping) is the unphysical creation of what Noh and Woodward termed [4] *flotsam* ("floating wreckage") and *jetsam* ("jettisoned goods"). These terms are appropriate for

[1] In a true sense DeBar's linear reconstruction is not finding a "normal" to the interface, but rather estimating a linear distribution of the volumes based solely on one-dimensional information.

isolated, submesh-size material bodies that separate from the main material body because of errors induced by the volume tracking algorithm. These material remnants tend to be ejected from interfaces in piecewise constant volume tracking methods when the flow has significant vorticity and/or shear near the interface. An example of this behavior is demonstrated clearly with the SLIC results presented later. The presence of flotsam near interfaces can severely compromise the overall interfacial flow solution, especially when interface dynamics (e.g., surface tension and phase change) are also being modeled.

2.3. *Piecewise Linear Volume Tracking Methods*

In the early 1980s, volume tracking methods were advanced significantly by the new piecewise linear schemes of Youngs and coworkers [9, 21]. Youngs' methods positioned each reconstructed interface line, defined by a slope and intercept, within the volume fraction control volume (cell). This is in contrast to DeBar's method, where the reconstructed interface was positioned across cell faces. The slope of the line is given by the interface normal (gradient of the volume fractions), and the intercept follows from invoking volume conservation. The interface normal is determined with a multidimensional algorithm (9-point stencil in two dimensions, 27-point stencil in three) that does not depend upon the sweep direction. Interestingly, Youngs' second method cannot reconstruct a volume distribution resulting from a line despite its choice of a linear basis in a cell. The methods of Youngs, formulated for both two [9] and three [21] dimensions on orthogonal meshes, were subsequently adopted in many high-speed hydrocodes involving material interfaces [22–25].

The two-dimensional (2D) and three-dimensional (3D) piecewise linear methods developed by Youngs differed by more than just dimensionality. Although the time integration scheme for both methods was identical (operator splitting), the normal used for interface reconstruction was more accurate in the 2D algorithm than the 3D algorithm. The interface normal computed in Youngs' 2D algorithm will reproduce a line, regardless of its orientation on an orthogonal mesh, and is, therefore, second-order accurate (according to Pilliod and Puckett's criteria [10, 26]). The 3D normal will reproduce a plane for certain simple orientations; hence, the algorithm is not formally second-order accurate. We will refer to Youngs' 2D and 3D methods as "Youngs' first method" and "Youngs' second method," respectively, because of this important accuracy difference. It is obviously desirable to retain second-order accuracy in a piecewise linear volume tracking method, otherwise a line (or plane in 3D) will not be preserved after simple translation.

Many extensions and enhancements to the significant work of Youngs have occurred since its introduction. Johnson extended Youngs' 2D method to nonorthogonal meshes [22]. The first use adaptive mesh refinement (AMR) in a volume tracking method can be found in [27]. Puckett and Saltzman [26, 28] coupled an AMR algorithm [29, 30] to the 3D method. Pilliod and Puckett have recently refined Youngs' algorithm in two dimensions with an unsplit, "corner-coupled" time integration scheme extension that has second-order accuracy through the use of an improved interface normal [10, 11]. A similar method is used in [15] for comparison with other interface tracking methodologies. Mosso [31] has recently introduced new methods for second-order interface normal approximations on irregular meshes and devised a new second-order time integration scheme based on the concept of remapping a displaced mesh. This approach couples nicely with arbitrary Lagrangian–Eulerian (ALE) schemes. Finally, Kothe and coworkers have extended

Youngs' 3D method to unstructured meshes [32] by introducing a second-order Runge–Kutta method for time integration and a robust method for plane truncation of arbitrary polyhedra.

The more recent FLAIR method [8] is similar to DeBar's method, in that piecewise linear interfaces are reconstructed across cell faces rather than within a volume bounded by cell faces. The face-centering of piecewise linear interfaces in the FLAIR method results in a reconstructed interface that can be more continuous than a cell-centered reconstruction. Since the FLAIR time-integration scheme is not presented in detail, however, direct comparisons with the current method are not made in this paper.

We will refer to the family of piecewise methods introduced by Youngs (and its extensions) as PLIC (for piecewise linear interface calculation) methods. Details and capabilities of many PLIC volume tracking methods unfortunately remain obscure because of insufficient widespread publication. Despite this fact, PLIC methods have been used successively for high speed hydrodynamic calculations this past decade by a host of researchers. Progress is currently being made by the authors [33, 34] and others [35] toward applying PLIC to incompressible multiphase flows.

As demonstrated later, the PLIC piecewise linear interface approximation, coupled with a second-order time integration scheme, results in a second-order algorithm that clearly outperforms the above-mentioned classes of first-order piecewise constant methods. Despite the notable improvements offered by PLIC, variants of piecewise constant (SLIC) and piecewise constant/stair-step (VOF) methods have remained in widespread use because of their simple implementation.

An important contribution in the SLIC method was a prescription for cells containing multiple materials. As an example, consider the actual three-material configuration in Fig. 4. Most volume tracking methods model this situation by first reconstructing the material one/two–three interface (by combining material two and three volume fractions), then the one–two/three interface (by combining the material one and two volume fractions). For example, the reconstructed SLIC and PLIC interfaces for this configuration are shown in Fig. 4. The approximation will unfortunately yield reconstructed interfaces that are positioned correctly only in the simplest of cases. This "onion-skinning" approach for reconstructing multiple interfaces can unfortunately lead to interface intersections, which causes fluid volume to be incorrectly associated with more than one material. This intersection problem is more acute with PLIC methods. Another problem inherent to the onion-skin model is the required ordering (prioritizing) of material volume fluxes at cell boundaries. Multiple material models superior to these onion-skin models must be devised before volume tracking methods can reliably model complex topology multiple-material flows.

We now derive the governing volume evolution equations. The evolution equations are trivial for incompressible flows, but require modeling assumptions for compressible flows.

3. FLUID VOLUME EVOLUTION EQUATIONS

We first derive material volume evolution equations in the presence of an incompressible flow field. This is a straightforward manipulation of standard consistency relations. We begin by defining V_k, the volume of material k,

$$V_k = \int \alpha_k(V)\, dV, \tag{1a}$$

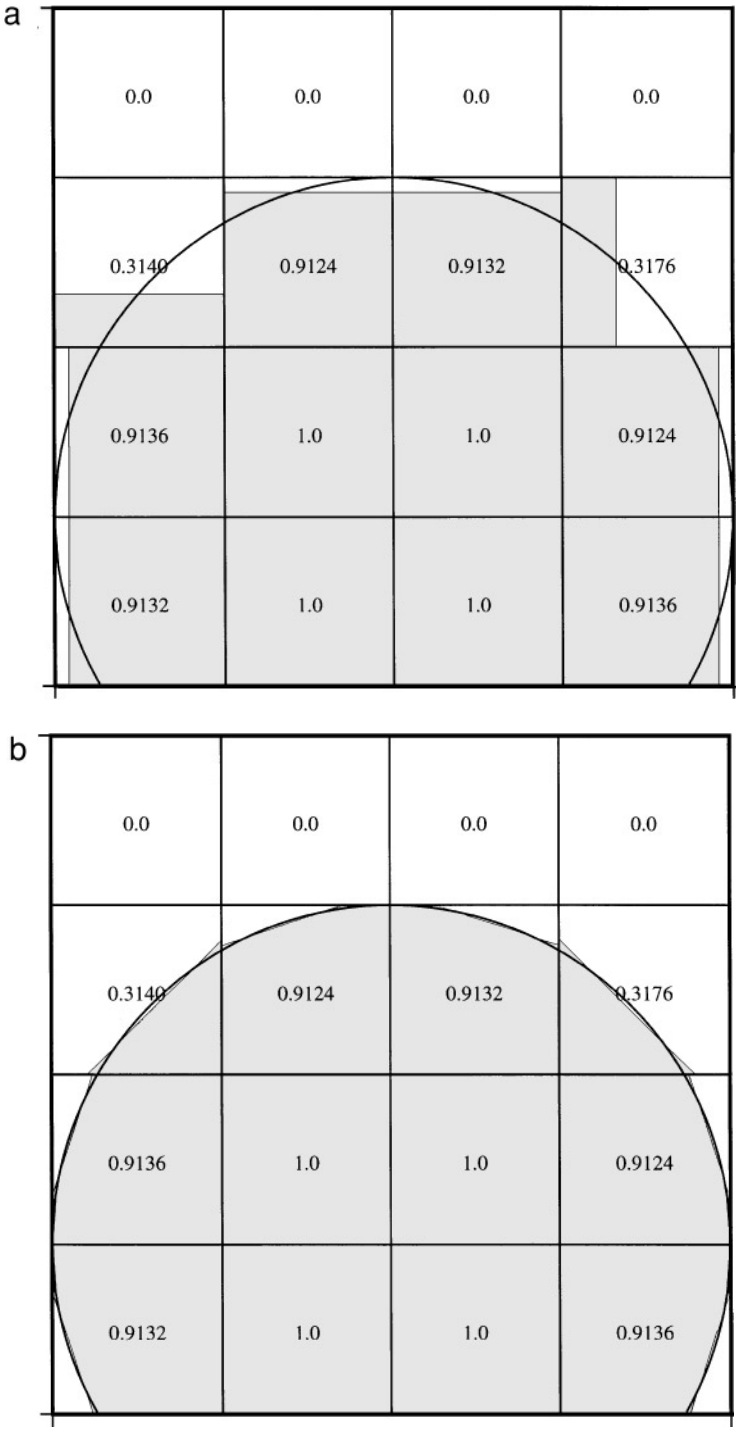

FIG. 3. Reconstructed interfaces (shaded regions) for a circle (continuous line) using the SLIC and PLIC methods. The piecewise constant approximation in SLIC forces the reconstruction to align with selected mesh logical coordinates, whereas the piecewise linear approximation in PLIC allows the reconstruction to align naturally with the interface. Numbers in the cells denote volume fractions. (a) SLIC reconstruction. (b) PLIC reconstruction.

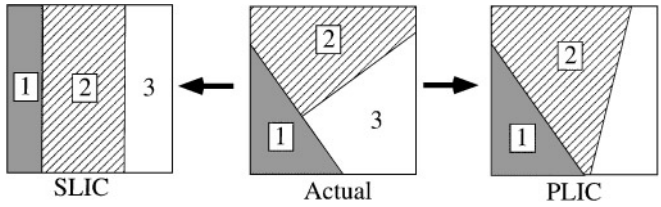

FIG. 4. Volume-tracking methods traditionally model multiple materials with an onion-skin algorithm. Shown in the middle is an example of three-material configuration, with SLIC and PLIC representations shown on the left and right, respectively. It should be noted that if care is not taken with the PLIC method, the linear interfaces can overlap in an unphysical manner.

where $\alpha(V)$ is an indicator function given by

$$\alpha_k(V) = \begin{cases} 1, & \text{if fluid } k \text{ is present;} \\ 0, & \text{otherwise.} \end{cases} \tag{1b}$$

Given V_k, the volume fraction f_k is defined as

$$f_k = \frac{V_k}{V}, \tag{2}$$

where the total volume V is $\int dV$. We require that the material volumes fill all space,

$$V = \sum_k V_k,$$

or, equivalently, $\sum_k f_k = 1$.

It is readily apparent in the following that volume tracking methods are naturally control volume methods and the volume fractions f (dropping the k subscript) are integrally averaged quantities. Given a flow field $\mathbf{u}$, a standard advection equation governs the evolution of f,

$$\frac{df}{dt} = 0 \rightarrow \frac{\partial f}{\partial t} + \mathbf{u} \cdot \boldsymbol{\nabla} f = 0. \tag{3a}$$

If the flow field is incompressible, i.e., $\nabla \cdot \mathbf{u} = 0$, the f advection equation can be recast in conservative form:

$$\frac{\partial f}{\partial t} + \mathbf{u} \cdot \boldsymbol{\nabla} f \rightarrow \frac{\partial f}{\partial t} + \nabla \cdot (\mathbf{u} f) = 0. \tag{3b}$$

An equivalent statement for (3b) is that material volumes remain constant on streamlines. Incompressibility allows this statement to be further expressed as a conservation law. This confirms our intuition that incompressible flow conserves volume, and it also conserves the apportioning of material volumes into the total volume (except for any boundary flux).

4. METHOD OF SOLUTION

We now describe our algorithm for the geometric solution of the volume evolution equations given in Section 3. Our algorithm appeals to geometry because material volume fluxes,

defined as material volumes passing through a given cell face over one time step, are the n-sided polygons formed by interface line segments passing through total volume flux polygons. These fluxes are computed in a straightforward and systematic manner using algorithms for lines intersecting n-sided polygons. Our algorithms are constructed from a "geometric toolbox," as described in Section 4.1. By using this toolbox, heuristic, "case-by-case" logic is not required to find solutions. We also discuss our unsplit method for the time integration of the volume evolution equations.

After presenting our geometric toolbox in Section 4.1, we describe the piecewise linear interface reconstruction in Section 4.2 (details are given in Appendix A), material volume flux computation in Section 4.3, and time integration in Section 4.4. Examples of the method's performance on basic translation and rotation tests can be found in Appendix B.

4.1. *A Geometric Toolbox*

Our 2D PLIC method requires the following geometric functions:

1. Line–line intersection
2. Point location
3. Polygon collection
4. Polygon area.

These functions are simple, well-defined, and widely used, for example, in the field of computational geometry [36]. Concise algorithm design and implementation is made possible with these functions, as is evident later in this section.[2] The capabilities of this toolbox can also be extended to accommodate a 3D PLIC method if additional functions such as plane/surface intersection are incorporated [32].

Our geometric functions assume that a line is defined by the equation

$$\mathbf{n} \cdot \mathbf{x} + \rho = 0, \tag{4}$$

where $\mathbf{n}$ is the normal to the line, $\mathbf{x}$ is a point on the line, and ρ is the line constant. Computational cells are defined (in 2D) as n-sided polygons given by a set of n vertices $\mathbf{X}_v = (x_v, y_v)$. Any cell having volume fractions f between zero and one will possess an interface defined by (4). The interface line equation will in general be different in each interface cell; i.e., values of $\mathbf{n}$ and ρ will vary (from cell to cell) since the overall interface geometry is approximated as piecewise linear. As discussed later in this section, values of $\mathbf{n}$ and ρ in (4) result from a volume fraction gradient and enforcement of volume conservation, respectively. For each interface cell, the interface line divides space into regions inside the fluid and outside the fluid, depending upon the convention chosen for $\mathbf{n}$. We choose $\mathbf{n}$ to point into the fluid; hence, (4) will be positive for any point $\mathbf{x}$ lying within the fluid, zero for any point $\mathbf{x}$ lying on the line, and negative for any point $\mathbf{x}$ lying outside of the fluid.

Below we summarize each of the four geometric functions.

Function 1: Line–line intersection. The most basic geometric function is one that locates the point of intersection between two lines. If the lines are actually line segments, as in our PLIC method, steps must be taken to determine if the intersection point is valid (i.e., lies within segments). The intersection point is found from a simultaneous solution of the two line equations (with checks for parallel lines). By invoking this function only in cases

[2] The source code for our geometric toolbox is available at `http://lune.mst.lanl.gov/Telluride/Text/publications.html`.

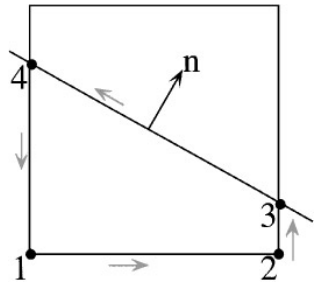

FIG. 5. Typical example of a 4-sided polygon formed when an interface line truncates a computational cell. Functions 1–4 in the text are performed sequentially to compute the area surrounded by this polygon. Arrows indicate the line integration path.

where there must be a valid intersection point, costly checks for validity are avoided. A valid intersection point will result when one end of one line segment lies "across" the intersection point. This function is needed to find the points of intersection between the interface line and any cell edge.

Function 2: Point location. Given a line defined by (4), a point location function returns true if a point (x_v, y_v) lies inside the fluid, which is true if (4) is positive for (x_v, y_v). Given a point (x_i, y_i) on the interface, the point (x_v, y_v) lies within the fluid if

$$n_x(x_v - x_i) + n_y(y_v - y_i) > 0.$$

This function is needed to determine which of the cell vertices $\mathbf{X}_v$ lie inside the fluid.

Function 3: Polygon collection. A polygon collection function collects the vertices (x_v, y_v) of a n-sided polygon in counterclockwise order. The polygon vertices collected for each interface cell are those cell vertices lying inside the fluid and interface line/cell edge intersection points. A 4-sided polygon example is shown in Fig. 5. By definition, two adjacent cell vertices are on opposite sides of the interface line if one cell vertex lies inside the fluid and the other lies outside. The resultant n-sided polygon surrounds the fluid in that cell, as seen in Fig. 5. This function is needed before the area inside the n-sided polygon can be computed.

The polygon collection function uses the output of Function 2, which initializes a boolean variable that describes the "state" of cell vertices with respect to the interface (i.e., inside or outside the fluid). A vertex state variable minimizes source code logic and enables robust procedures. The benefits of this implementation are easily realized for the (fairly typical) case of an interface line passing nearby a cell vertex (see Fig. 6). In this case a decision regarding the identity of this cell vertex is forced (based on some prescribed tolerance), thereby avoiding numerical difficulties. This problem and its solution are illustrated by the two relevant cases in Fig. 6. The cost of cell vertex ambiguity situations is an extra vertex that becomes associated with the truncated polygon.

Function 4: Polygon area. A polygon area function takes the vertices (x_v, y_v) of an n-sided polygon, collected in counterclockwise order (the result of Function 3) and computes the exact area enclosed by the polygon. In Cartesian geometry, the area enclosed by the polygon is given by

$$A = \frac{1}{2}\sum_{v=1}^{n}(x_v y_{v+1} - x_{v+1}y_v), \tag{5}$$

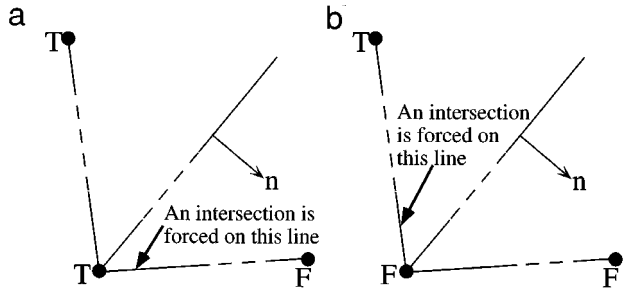

FIG. 6. Two examples of an interface line passing nearby a cell vertex. This vertex is "ambiguous" in that it can be considered inside or outside the fluid, depending upon the prescribed tolerance of the boolean variable that identifies the vertex as being inside or outside the fluid. In these figures, "T" means that the vertex is inside the interface (true) and "F" means that the vertex is outside the interface (false). (a) The critical vertex is inside the fluid. (b) The critical vertex is outside the fluid.

where vertex $v = n + 1$ is assumed to coincide with vertex $v = 1$. In cylindrical geometry having azimuthal (θ) symmetry, the area (an azimuthally-symmetric volume) is given by

$$A = \frac{\pi}{6}\sum_{v=1}^{n}(r_v + r_{v+1})(r_v z_{v+1} - r_{v+1} z_v). \tag{6}$$

The single algorithmic change required for our method to perform correctly in 2D cylindrical rather than Cartesian geometry is the use of (6) instead of (5) for the polygon area computation.

4.2. *Reconstructing the Interface*

Given the functions provided by our geometric toolbox, linear interface segments must be *reconstructed* in each mixed cell. This reconstruction step requires line equation (4) to be defined for each interface segment, so an interface normal **n** and line constant ρ must be determined. The line constant ρ follows from enforcement of volume conservation, and the interface normal **n** follows from volume fraction gradients. Interface reconstruction examples of simple volume fraction distributions (circles and squares) are given in Appendix A. The examples in particular illustrate the importance of an accurate, linearity-preserving estimate for **n**.

Determining the line constant ρ is the most difficult reconstruction task because the value of ρ is constrained by volume conservation. In other words, the value of ρ is constrained such that the resulting line passes through the cell with a truncation volume equal to the cell material volume V. This determination requires inverting a $V(\rho)$ relation, in which V can vary linearly, quadratically, or cubically with ρ, depending upon the coordinate system and the shape of the n-sided polygon formed by the interface segment truncating the cell. It is tempting to construct an algorithm for determining ρ based on a direct solution of the $V(\rho)$ relation. But, since this relation is often nonlinear and varies in each mixed cell, due to its dependence upon local data, a "case-by-case" implementation results that is not efficient (vectorizable or parallel), general, concise, or easily maintained and understood. We instead choose to invert the $V(\rho)$ relation iteratively in each mixed cell. The resulting algorithm procedural logic is therefore independent of the mixed-cell properties and data. As implemented, the algorithm is simple, robust, general, efficient, and easily understood.

The line constant ρ is found when the generally nonlinear function

$$f(\rho) = V(\rho) - V,$$

becomes zero. Here $V(\rho)$ is the material volume in the cell bounded by the interface segment (with line constant of ρ) and the portion of the cell edges within the material. When these two volumes are equal (to within some tolerance), the interface segment is declared "reconstructed" in that cell. A host of root-finding algorithms are available to find the zero of this function, but we have found Brent's method [37, Chap. 9] to give the best results in practice. Bisection will converge, but is slow, and Newton's method may diverge, but Brent's method invokes a combination of bisection and inverse quadratic interpolation to find a near-optimal next guess for ρ. This approach is well suited for our $V(\rho)$ function since V often varies quadratically with ρ (see Fig. 8). Newton's method typically converges in half the iterations of Brent's method, but Brent's method requires less computational effort because an evaluation of the derivative of V with respect to ρ is not needed.

An intelligent initial guess for ρ aids in efficient convergence (<4–6 iterations). We currently initialize ρ prior to the iteration in the following manner. Lines possessing the interface normal **n** are passed through each vertex of the polygonal cell, and the resulting truncation volumes are computed. Those two lines forming truncation volumes that bound the actual material volume in that cell provide upper ($\rho_{\max}$) and lower ($\rho_{\min}$) bounds for ρ. Our initial guess for ρ is then a linear average of $\rho_{\min}$ and $\rho_{\max}$.

Figure 7 illustrates the iterative progression of Brent's algorithm in finding ρ for an example reconstruction. We summarize with an algorithm template for finding the interface constant ρ:

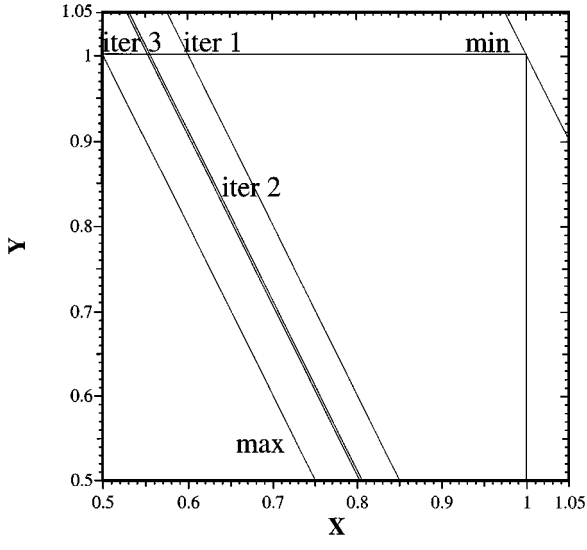

FIG. 7. An example of the iterative placement of an interface segment within a mixed cell using Brent's method, which prescribes a new value for interface line constant ρ each iteration. The line is properly placed after five iterations, although only the first three are shown here (after which progress is indistinguishable). Initial bounding guesses ($\rho_{\min}$ and $\rho_{\max}$) for the line are also shown. This example corresponds to the $f = 0.2$ case in Fig. 9b.

Interface Reconstruction Template

1. Given an interface line segment, described by Eq. (4), truncate a mixed cell.
2. Find and assemble the n vertices of the polygon formed by the those cell vertices inside the fluid and the interface line/cell edge intersection points.
3. Compute the volume bounded by this polygon.
4. Determine if the polygon volume differs from the known fluid volume by some prescribed tolerance.
5. If the volumes differ, use Brent's method to find a new estimate for ρ in Eq. (4) and go back to step 1.
6. If the volumes do not differ, the interface line is declared reconstructed.

End template

If this template is applied only to mixed cells, an iterative search for ρ is computationally efficient, focusing efforts only where necessary. This approach is also attractive because mixed cells usually comprise $<10\%$ of the total cells in the computational domain, even for the most topologically complex cases. Also, because volume tracking methods are local in nature, the increase in mixed cell number over a given time step is bounded because only those cells that are mixed or directly adjacent to mixed cells experience a volume fraction change. An example of the behavior of this algorithm is shown in Fig. 8, where the line constant ρ and iteration count as a function of fluid volume are shown for a given interface normal **n**. The actual interface line placements are shown in Fig. 9.

4.3. *Material Volume Fluxes*

Integration of the material volume evolution equations discussed in Section 3 requires the evaluation of material volume fluxes δV at each cell face. The fluxes δV represent the volume of material passing through a given face during the current time step. The sum of all material volume fluxes must equal the total volume flux, which for face $(i + \frac{1}{2}, j)$ is

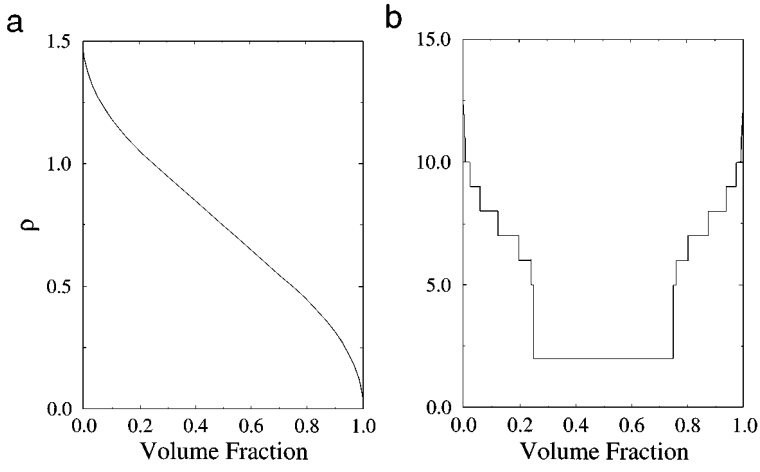

FIG. 8. An example of the variation of line constant ρ and Brent's method iteration count for a given interface normal **n**. Note the linear and quadratic dependence of ρ as a function of fluid volume. The actual interface line placements for a number of cases are given in Fig. 9. (a) Line constant ρ as a function of fluid volume. (b) Interface reconstruction iteration count as a function of fluid volume.

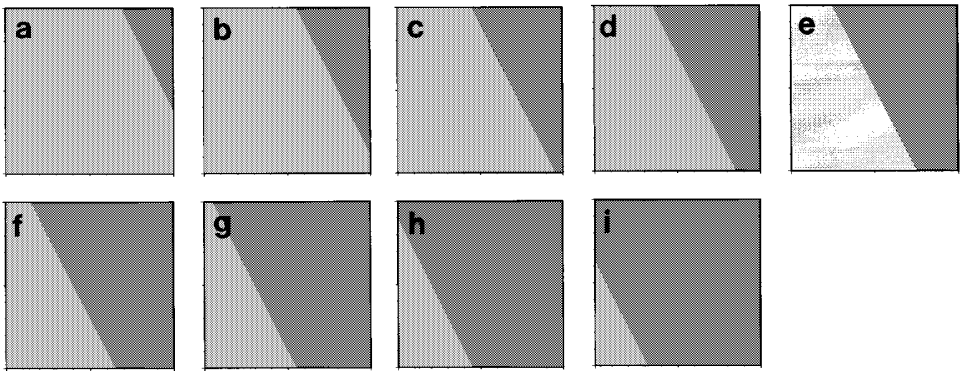

FIG. 9. Interface line placements for the case shown in Fig. 8. Darker regions correspond to the fluid whose interface is being reconstructed. (a) $f = 0.1$. (b) $f = 0.2$. (c) $f = 0.3$. (d) $f = 0.4$. (e) $f = 0.5$. (f) $f = 0.6$. (g) $f = 0.7$. (h) $f = 0.8$. (i) $f = 0.9$.

given by

$$\sum_k \delta V^k_{i+1/2,\,j} = \delta V_{i+1/2,\,j} = \Delta t\, \mathbf{u}_{i+1/2,\,j} \cdot \mathbf{A}_{i+1/2,\,j}, \tag{7}$$

where **u** is the velocity vector and **A** is the edge area vector.

To compute δV, two steps are required: (1) the polygons bounding the volume swept by the velocity field over the time step must be constructed, and (2) the amount of material k must be found for these polygons. The machinery to accomplish the second step has been discussed earlier (the geometric toolbox given in Section 4.1), but step 1 requires definition.

Here we develop both operator split and unsplit time integration schemes, which are depicted schematically in Fig. 10. The operator split method constructs the solution in a series of one-dimensional sweeps shown in Figs. 10a, b. Usually Strang splitting [1] is employed to lessen the symmetry breaking effects of the operator splitting. A naive unsplit method is shown in Fig. 10c. This method has undesirable overlapping volume flux regions. In our unsplit method (Fig. 10d), the flow is considered to be completely multidimensional, but the method can be constructed from the operator split method via systematic corrections shown below.

As an example, consider the formation of edge flux volumes for an x-sweep of an operator split method. A sample result is shown in Fig. 11a. The steps to form the edge flux polygon for face $(i + \frac{1}{2}, j)$ are outlined in the following template.

Edge Flux Polygon Template

1. Identify the vertices of the edge being evaluated, which form the first two vertices in the flux polygon: $(x_{i+1/2,\,j-1/2}, y_{i+1/2,\,j-1/2})$, $(x_{i+1/2,\,j+1/2}, y_{i+1/2,\,j+1/2})$.
2. Given the velocity normal to this edge $(u_{i+1/2,\,j})$ and the time step (Δt), trace characteristically to locate the final vertices: $(x_{i+1/2,\,j-1/2} - u_{i+1/2,\,j}\Delta t, y_{i+1/2,\,j-1/2})$, $(x_{i+1/2,\,j+1/2} - u_{i+1/2,\,j}\Delta t, y_{i+1/2,\,j+1/2})$.

End template

The unsplit algorithm requires that corrections be made to the edge (operator split) flux volume defined above. The corrections are defined by the velocity field assumed in the algorithm. In our implementation, the corner velocity transverse to the direction of the

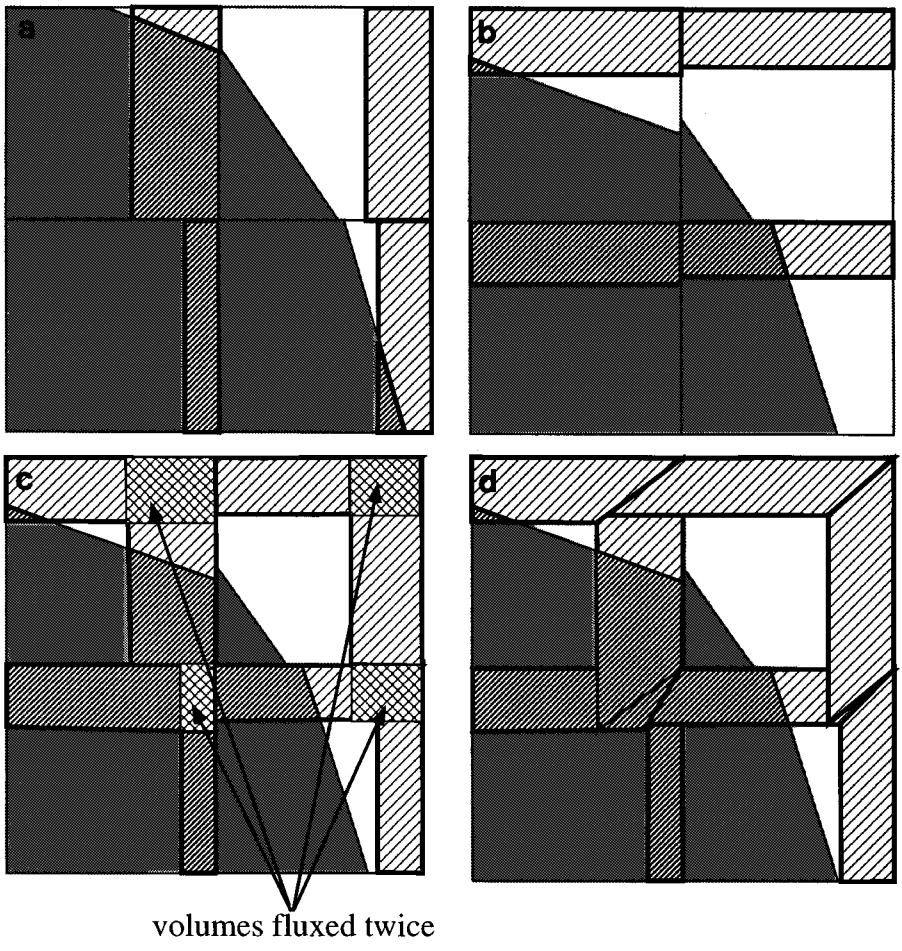

FIG. 10. Face volume fluxes δV and their bounding polygons for three different time-integration schemes in two dimensions. For operator-split integration, an x-direction sweep is made in (a), followed by a y-direction sweep in (b). A naive unsplit method, shown in (c), is not recommended because face-adjacent polygons overlap, hindering conservative, monotonic volume advection. In our unsplit method of (d), the polygons are constructed by tracing along upwind characteristics, yielding a multidimensional, "corner-coupled" fluxing of material volumes. Problems with an unsplit integration scheme can necessitate redistribution algorithms as shown in Fig. 12. (a) An x-direction sweep of an operator-split method. (b) A y-direction sweep of an operator-split method. (c) A naive multi-dimensional unsplit method. (d) The unsplit multi-dimensional method.

edge under consideration is "upwinded" given the sign of the normal edge velocity. For example if $\mathbf{u}_{i+1/2,j} > 0$, the velocities $v_{i,j\pm1/2}$ will define the corner corrections to the flux volume at the $(i + \frac{1}{2}, j)$ edge. An example corner correction polygon is depicted in Fig. 11b. The correction polygons are computed according to the following corner flux polygon template.

Corner Flux Polygon Template

1. Identify the vertex of the corner being evaluated, which forms the first vertex in the flux polygon: $(x_{i+1/2,j+1/2}, y_{i+1/2,j+1/2})$.
2. Using the velocity normal to this edge $(u_{i+1/2,j})$ and the time step (Δt), trace characteristically to the next vertex: $(x_{i+1/2,j+1/2} - u_{i+1/2,j}\Delta t, y_{i+1/2,j+1/2})$.

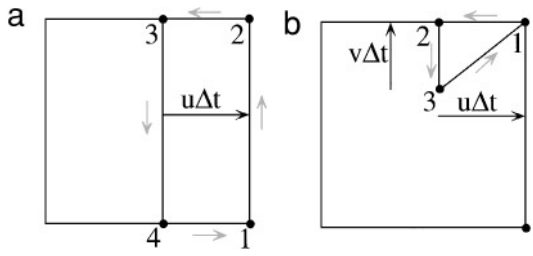

FIG. 11. Edge and corner polygons formed at face $(i + \frac{1}{2}, j)$ when tracing back along a positive velocity field. The edge polygon that encloses volume flux $\delta V_{i+1/2,j}$ is defined by the four points in (a) for the x-sweep of an operator-split time integration. For multidimensional unsplit time integration, the edge polygon in (a) must be modified by corner fluxes such as the triangle in (b). (a) Quadrilateral edge volume flux polygon. (b) Triangular corner volume flux polygon.

3. Using the upwinded transverse velocity at the corner, trace characteristically to the final vertex: $(x_{i+1/2,j+1/2} - u_{i+1/2,j}\Delta t, y_{i+1/2,j+1/2} - v_{i+1/2,j+1/2}\Delta t)$. The value of $v_{i+1/2,j+1/2}$ is determined by the sign of $u_{i+1/2,j}$. For example, if $u_{i+1/2,j} > 0$, then $v_{i+1/2,j+1/2} = v_{i,j+1/2}$.

End template

Given the edge and corner flux polygon templates, unsplit edge flux polygons at face $(i + \frac{1}{2}, j)$ are found according to the following template.

Unsplit Edge Flux Polygon Template

1. Apply the *edge flux polygon template*.
2. Apply the *corner flux polygon template* at $(i + \frac{1}{2}, j + \frac{1}{2})$.
3. If the upwinded transverse velocity at $(x_{i+1/2,j+1/2}, y_{i+1/2,j+1/2})$ is positive, subtract the corner flux volume; otherwise add it.
4. Apply the *corner flux polygon template* at $(i + \frac{1}{2}, j - \frac{1}{2})$.
5. If the upwinded transverse velocity at $(x_{i+1/2,j-1/2}, y_{i+1/2,j-1/2})$ is positive, add the corner flux volume; otherwise subtract it.

End template

Next, we discuss our temporal integration methods.

4.4. *Time Integration*

We now consider the time integration of (3b) for incompressible flows, whereby the material k volume fractions $f^{k,n}$ at discrete time level n are marched forward in time to step $n + 1$. In performing this time integration, care must be taken to ensure that material volumes V_k remain volume-filling in each cell; i.e., material volumes must sum to the total cell volume $V(= \sum_k V_k)$. For incompressible flow, an additional global volume-filling constraint applies, namely material volume summed over the entire mesh must also be conserved (this applies locally as well). For compressible flows, any local discrete volume-filling errors can be absorbed by the fluid compressibility, but for incompressible flow, this can lead to unacceptable errors. If compressibility (velocity divergence) is used to absorb local volume-filling errors for incompressible flows, the integration accuracy of the right-hand side of (3b) degrades accordingly.

A key point in our time integration scheme for the incompressible flow volume evolution equations is the recognition that discrete velocity divergences are not necessarily zero, but

rather a function of the convergence tolerance used in obtaining the linear system solutions. To employ this "divergence correction," $\nabla \cdot \mathbf{u}$ is added to both sides of (3b), giving

$$\frac{\partial f}{\partial t} + \nabla \cdot (\mathbf{u} f) = f \nabla \cdot \mathbf{u}. \tag{8}$$

By applying this correction, even for incompressible flows, we find that local and global volume-filling constraints are adhered to much more closely. We integrate (8) forward in time to advance volume fractions from $f^{k,n}$ to $f^{k,n+1}$ in the presence of incompressible flows.

Our time integration schemes operate on two basic sets of cells: those that are *mixed* and those that are *active*. Before integration proceeds, those cells defined as mixed and active are flagged appropriately. Mixed cells are labeled as such if:

1. Their volume fractions f satisfy $\epsilon \le f \le 1 - \epsilon$; or
2. Their volume fractions f satisfy $f > 1 - \epsilon$; *and* a face is shared with an empty cell having $f < \epsilon$.

Active cells are labeled as such if at least one cell in their domain of dependence is mixed. In two dimensions, the domain of dependence for operator-split time integration is three cells along the current sweep direction, while it is a 3×3 array of cells for unsplit time integration. In flagging mixed and active cells, ϵ is a small number relative to zero, taken to be 1×10^{-12} for this work (appropriate for double precision arithmetic). This cell categorization localizes algorithm computational work in the proximity of interface. For 2D computations of topologically simple interfaces, we have found the work required to time-integrate the volume evolution equations scales like the square root of the total grid size.

For an operator-split time integration scheme, (8) must be integrated twice in two dimensions (thrice in three dimensions), one integration per each sweep, and one sweep per dimension. The volume fractions $f^{k,n+1}$ are therefore constructed from multiple, sequential solutions to (8). In two dimensions, volume fractions $f^{k,n}$ are advanced in the first sweep to $\tilde{f}$ according to

$$\begin{aligned} \tilde{f}_{i,j} = f^{k,n}_{i,j} &- \frac{\delta V_{i+\delta i, j+\delta j} - \delta V_{i-\delta i, j-\delta j}}{V_{i,j}} \\ &+ \Delta t f^{k,n}_{i,j} \frac{A_{i+\delta i, j+\delta j} \mathbf{u}_{i+\delta i, j+\delta j} - A_{i-\delta i, j-\delta j} \mathbf{u}_{i-\delta i, j-\delta j}}{V_{i,j}}, \end{aligned} \tag{9}$$

and in the second sweep $\tilde{f}$ are advanced to their final values $f^{k,n+1}$:

$$\begin{aligned} f^{k,n+1}_{i,j} = \tilde{f}_{i,j} &- \frac{\delta V_{i+\delta i, j+\delta j} - \delta V_{i-\delta i, j-\delta j}}{V_{i,j}} \\ &+ \Delta t f^{k,n+1}_{i,j} \frac{A_{i+\delta i, j+\delta_j} \mathbf{u}_{i+\delta i, j+\delta j} - A_{i-\delta i, j-\delta j} \mathbf{u}_{i-\delta i, j-\delta j}}{V_{i,j}}. \end{aligned} \tag{10}$$

Since our method is implemented on structured, orthogonal meshes in Cartesian (or azimuthally symmetric cylindrical) geometry, each sweep is associated with either an x (r) or y (z) direction. For sweeps in the x (r) direction, $(\delta i, \delta j) = (1/2, 0)$, and for sweeps in the y (z) direction, $(\delta i, \delta j) = (0, 1/2)$. Sweep direction order is alternated every time step to minimize asymmetries induced by the sequential sweeping process.

Note the RHS rightmost terms in (9) and (10), which are the important divergence correction terms. These terms, needed to enforce volume-filling constraints, contain volume

fractions f at differing time levels, explicit (time level n) for the first sweep and implicit (time level $n+1$) for the second sweep. This form of the correction is found to be optimal in practice, giving a net divergence correction that employs a volume fraction having an intermediate time level. We now summarize our operator-split time integration template.

Operator-Split Time Integration Template

1. Flag all mixed, active, and isolated cells.
2. Compute the discrete velocity divergence $\nabla \cdot \mathbf{u}_{i,j}$ in all flagged cells.
3. Reconstruct interfaces in all mixed cells according to the Interface Reconstruction Template given in Section 4.2.
4. Compute edge volume fluxes δV in all flagged cells according to the Edge Flux Polygon Template given in Section 4.3. If this is an x (r) direction sweep, the fluxes will be right-face fluxes; if this is a y (z) direction sweep, they will be top-face fluxes.
5. Advance volume fractions f in time using (9) if this is the first sweep or (10) if this is the second sweep.
6. Look for and conservatively redistribute any volume fraction undershoots ($f < 0$) or over shoots ($f > 1$).

End template

As a final note, regarding operator-split time integration, if $\nabla \cdot \mathbf{u} = 0$ *is* assumed, then employing the identity (for 2D Cartesian geometry)

$$\frac{\partial u}{\partial x} = -\frac{\partial v}{\partial y}$$

is often useful, especially for operator-split time integration discretizations of (3b). This approach appears to improve the discrete conservation properties of operator-split incompressible flow time integration [11, 35].

As opposed to an operator-split time integration of (8), an unsplit time integration scheme advances time level n volume fractions $f^{k,n}$ to $f^{k,n+1}$ with one equation, given for two dimensions as

$$f_{i,j}^{k,n+1} = f_{i,j}^{k,n} - \frac{\delta V_{i+1/2,j} - \delta V_{i-1/2,j}}{V_{i,j}} - \frac{\delta V_{i,j+1/2} - \delta V_{i,j-1/2}}{V_{i,j}} + \Delta t \frac{f_{i,j}^{k,n} + f^{k,n+1}}{2} \nabla \cdot \mathbf{u}_{ij}. \tag{11}$$

Note the centered time level $(n + 1/2)$ used for the volume fraction in the rightmost divergence correction term on the RHS above. We now summarize our unsplit time integration template.

Unsplit Time Integration Template

1. Flag all mixed, active and isolated cells.
2. Compute the discrete velocity divergence $\nabla \cdot \mathbf{u}_{i,j}$ in all flagged cells.
3. Reconstruct the interface in all mixed cells according to the Interface Reconstruction Template given in Section 4.2.
4. Compute volume fluxes δV in all flagged cells according to the Unsplit Edge Flux Polygon Template given in Section 4.3.
5. Advance volume fractions f in time using (11).
6. Look for and conservatively redistribute any volume fraction undershoots ($f < 0$) or overshoots ($f > 1$). The redistribution only acts on fluid in mixed cells, and in proportion to the fluid volume donating or receiving the redistributed fluid volume.

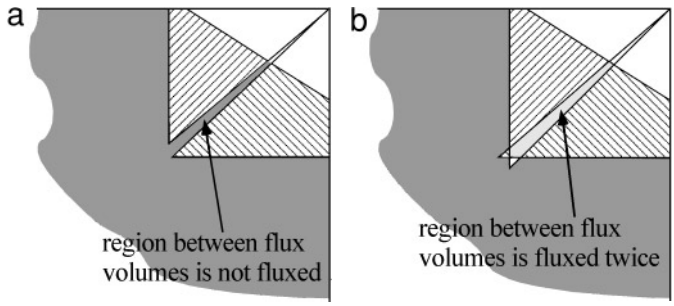

FIG. 12. Examples of two problems that can occur with the corner flux polygons in the presence of spatially varying velocity fields. (a) Fluid is not fluxed. (b) Fluid is fluxed twice.

End template

We have presented operator-split and unsplit algorithms for the time integration of (8), our volume evolution equation. These discretizations have assumed that the flow is incompressible. We illustrate the ability of our method to track fluid bodies in simple translational and rotational (hence, solenoidal) flow fields in Appendix B. Since these standard tests are frequently used in the literature to assess advection methods, these translation and rotation test results will facilitate comparison with other methods.

Problems can arise, however, when the velocity field possesses spatially varying vorticity. This spatial variation can sometimes lead to fluid being multiply fluxed or not fluxed at all. These subtle fluxing errors, which we identify and correct, are shown in Fig. 12 for the two most common instances. We currently correct these situations with a local redistribution algorithm. The tendency for these problems to occur is greater for algorithms possessing multidimensional time-integration schemes that do *not* incorporate the corner flux corrections to the edge fluxes (i.e., the method shown in Fig. 10c).

As long as the CFL condition is met, one-dimensional (edge) fluxes are without systematic problems. For velocity fields that vary in space, however, multidimensional fluxes possess small inconsistencies. Small volumes of fluid can be fluxed twice or not at all. As a consequence, there is a prospensity for methods based on multidimensional fluxes to produce small over/undershoots. This can be overcome through the use of a numerical velocity divergence definition,

$$\nabla \cdot \mathbf{u} = \sum_k \nabla \cdot (\mathbf{u} f^k), \tag{12}$$

which follows from the governing equation and ensures that the fractional volumes are volume filling. Equation (12) is not equivalent to the usual face-centered velocity divergence for the unsplit time integration because of the choice made for transverse velocities. This difference is present whenever the flow contains spatially varying vorticity.

5. RESULTS FOR VORTICAL FLOWS

Flows that induce simple translation or solid body rotation of fluid bodies do not adequately interrogate interface tracking methods designed for topology changes. Translation and rotation serve as useful debugging tests, but they are not sufficient for definitive analysis, in-depth understanding, or final judgement. Difficult (yet controlled) test problems having flows that bring about topology change elucidate algorithm strengths and weaknesses

relevant to modeling interfacial flows. Also, with controlled test problems, assessment of the interface tracking method is not obscured by subtleties of physics and algorithms in the flow solver.

Literature surveys indicate that uniform translation is the customary barometer for interface tracking methods. Solid body rotation tests accompany translation tests in more complete analyses. Typical examples of such tests can be found in [8, 20]. An acceptable tracking method must translate and rotate fluid bodies without significant distortion or degradation of fluid interfaces. Mass should also be converged rigorously in these cases. Translation and solid body rotation, however, enable only a *minimal* assessment of interface tracking algorithm integrity and capability because topology change is absent. Additional tests involving flows with nonuniform vorticity must be considered before a complete assessment can be made.

We therefore consider in this section two new 2D test problems that sufficiently challenge algorithm capabilities, provide meaningful metrics for measurement of algorithm performance, and are easy to implement. These problems, characterized by flows having nonconstant vorticity, were introduced recently in [15] as proposed metrics for any method designed to track interfaces undergoing gross topology change. Beside inducing topology change, the test problems are representative of interfacial flows in real physical systems, e.g., instabilities such as the Rayleigh–Taylor, Richtmeyer–Meshkov, and Kelvin–Helmholtz instabilities, where sharp gradients in fluid properties lead to vortical flow. A proper assessment of interface tracking methods should therefore impose strong vorticity at the interface.

Our test problems possess vortical flows that stretch and potentially tear any interfaces carried within the flow. The first problem contains a single vortex that will spin fluid elements, stretching them into a filament that spirals toward the vortex center. The flow field is taken from the "vortex-in-a-box" problem introduced in [38, 39]. The second problem has a flow field characterized by 16 vortices as introduced in [40]. This flow field causes fluid elements to undergo large topological changes. In the converged limit, fluid elements will not tear, instead forming thin filaments. The flow field in both problems is solenoidal, and is given cosinusoidal time-dependence following Leveque [41]. The temporal cosine term gives the flow the nice property of returning any fluid configuration to its initial state after one period. By forcing the flow to return to its initial state, quantitative comparison and evaluation can be simply computed with the help of error norms and convergence tests.

5.1. *Test Problems*

All test problems have identical initial conditions: a circle (radius 0.15) is centered at (0.50, 0.75) in a unit square computational domain. The domain is partitioned with either 32^2, 64^2, or 128^2 orthogonal, uniform cells. All boundaries are periodic. A scalar field is initialized to unity and zero inside and outside the circle, respectively. For those cells containing the circular interface, the scalar field is set to a value between zero and one, in proportion to the cell volume truncated by the circle. This field represents a characteristic (or color) function, which for our purposes is the fluid volume fraction for a circular fluid body, i.e., the volume fraction is 100% inside the circle and 0% outside.

Single vortex. A single vortex is imposed with a velocity field defined by the stream function [38],

$$\Psi = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y), \tag{13}$$

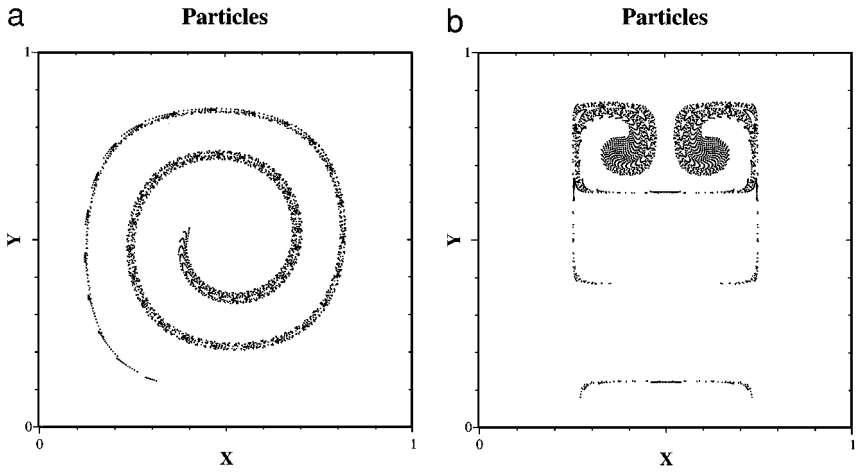

FIG. 13. (a) "Exact" solution at $t = 3$ for a circular fluid body placed in the single-vortex flow field on a 128^2 grid. (b) "Exact" solution at $t = 0.5$ for a circular fluid body placed in the deformation flow field on a 128^2 grid. In both cases the body is represented and tracked with analytical marker particles. (a) Single-vortex. (b) Deformation field.

where $u = -\partial\Psi/\partial y$ and $v = \partial\Psi/\partial x$. This is a solenoidal velocity field, which will deform bodies and promote topology changes. When the circular fluid body is placed in this field, it stretches and spirals about the center of the domain, wrapping around the center approximately two and a half times by $t = 3$, as shown in Fig. 13a. This represents an "exact" solution because fluid elements in the body for this case are represented with marker particles that move with the analytic velocity field given by (13) above. The marker particles are initialized in a uniformly spaced 4×4 array in each cell falling completely inside the circular fluid body. The 4×4 particle array is truncated in cells containing the circular interface and is absent in cells lying outside the circular interface.

Deformation field. A complex velocity field given by the stream function [40]

$$\Psi = \frac{1}{4\pi} \sin\left(4\pi\left(x + \frac{1}{2}\right)\right) \cos\left(4\pi\left(y + \frac{1}{2}\right)\right) \tag{14}$$

induces even more radical deformation and topology change of fluid bodies, providing a more stringent test than the field given by (13). This velocity field induces large deformation of the circular body by $t = 3$, as is evident in Fig. 13b, where results with the analytic particle method are shown. By this time, a large fraction of the circle has been entrained into the two nearest vortices, with a smaller portion (the thin filaments) having been entrapped by two nearby vortices.

Time-reversed flowfields. Following Leveque [41], the single vortex and deformation velocity fields can be multiplied by $\cos(\pi t/T)$, giving a flow that time-reverses (returns to its initial state) at $t = T$. In most of our tests we choose a period $T = 2$, hence the circular body will undergo large deformations until the first half period ($t = 1$), whereupon the flow will reverse, returning the circle to its initial undeformed state at the full period ($t = 2$). Error measurements are performed on the differences in data observed between $t = 0$ and $t = T$. These differences should ideally be zero, as the $t = 0$ and $t = T$ states should be identical.

5.2. *Results*

The performance of our volume-tracking method on these tests in assessed with both qualitative and quantitative fidelity measures. First, a qualitative assessment is made by comparing graphical results with the marker particle results shown previously. Second, a quantitative assessment is made with convergence results derived from error estimates. For the following tests, unless otherwise stated, we employ a CFL number of one (based on the maximum velocity in the domain) and use Pilliod's method for estimating the interface normal **n**.

Single vortex. This velocity field stretches and tears the initially circular fluid body as it becomes progressively entrained by the vortex. The entrainment is manifested as a long, thin fluid filament spiraling inward toward the vortex center. By integrating to late times, it is possible to observe the behavior of our volume-tracking method under rather extreme circumstances, whereby filaments become thinner than is supportable by the computational mesh. Convergence results are obtained by time-reversing the flow using Leveque's cosine term. As the reversal period T becomes longer, the fluid body evolves further away from its initial circular configuration; hence, it must undergo increasingly complicated topological change to reassemble properly at $t = T$.

Convergence results for the single vortex velocity field indicate that our method is remarkably resilient. The method exhibits second-order convergence, even for long periods ($T = 8$), where appreciable interface tearing and topological change has occurred. This is indicated by the L_1 error norms and convergence results shown in Table 2 for three different reversal periods. Figure 14 illustrates that solution errors (roughly a measure of phase error) are more evident for longer reversal periods. The solution quality, however, increases remarkably as the grid is refined (shown here for a 32^2 grid). Convergence is aided by the regularity of the velocity field, but the improvement with grid refinement is quite encouraging.

The under-resolved behavior of our PLIC method on the single-vortex problem is illustrated in Fig. 15. Here the solution becomes poor when interface topology is not resolved, i.e., as exhibited by single filament breaking into a series of fluid clumps that are each supportable by the reconstruction method. By $t = 3.0$ (Fig. 15d), the body has fragmented into numerous pieces. Despite the breakup, however, solution convergence does occur under grid refinement.

This behavior is reasonable and expected, given the assumptions inherent in the reconstruction, namely a piecewise linear interface approximation constrained by mass conservation. The breakup exhibited in Fig. 15 can be interpreted as an application of "numerical surface tension" along interfaces that are resolved inadequately. High curvature regions are

TABLE 2
L_1 Error Norms and Convergence Rates for a Circular Fluid Body Placed in the Time-Reversed, Single-Vortex Flow Field

Grid	Error ($T = 0.5$)	Order ($T = 0.5$)	Error ($T = 2.0$)	Order ($T = 2.0$)	Error ($T = 8.0$)	Order ($T = 8.0$)
32^2	7.29×10^{-4}		2.36×10^{-3}		4.78×10^{-2}	
		2.36		2.01		2.78
64^2	1.42×10^{-4}		5.85×10^{-4}		6.96×10^{-3}	
		1.86		2.16		2.27
128^2	3.90×10^{-5}		1.31×10^{-4}		1.44×10^{-3}	

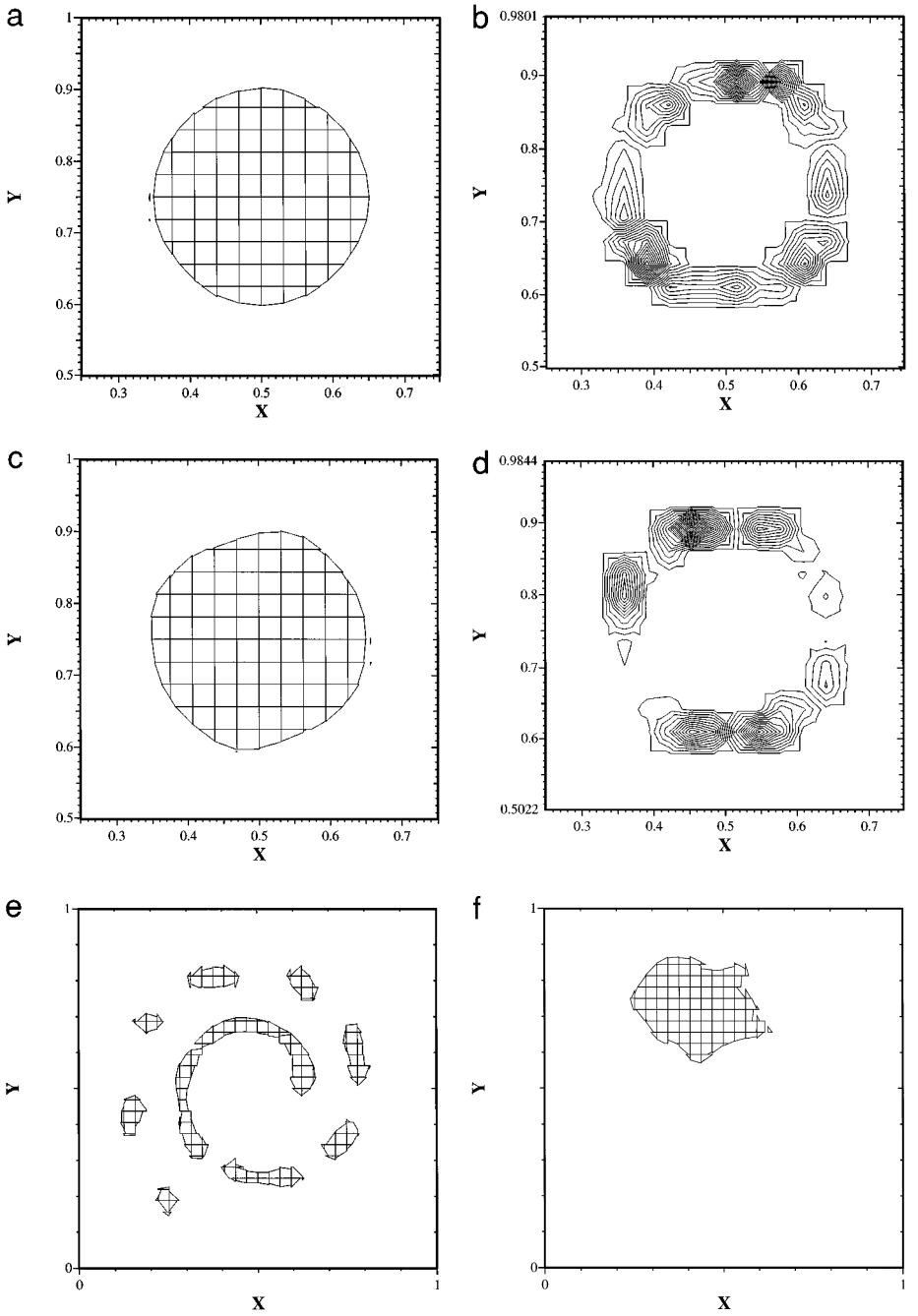

FIG. 14. Results for a circular fluid body placed in the time-reversed, single-vortex flow field on a 32^2 grid. (a) Reconstructed interfaces at $t = T$ for $T = 0.5$. (b) L_1 error contours at $t = T$ for $T = 0.5$. (c) Reconstructed interfaces at $t = T$ for $T = 2.0$. (d) L_1 error contours at $t = T$ for $T = 2.0$. (e) Reconstructed interfaces at $t = T/2$ for $T = 8.0$. (f) Reconstructed interfaces at $t = T$ for $T = 8.0$.

those unresolvable regions having interfaces with a radii of curvature less than roughly a mesh spacing. The piecewise linear interface approximation immediately flattens these regions, effectively applying numerical surface tension. Thin filament regions can also be the recipients of numerical surface tension, because poor linear reconstructions occur in

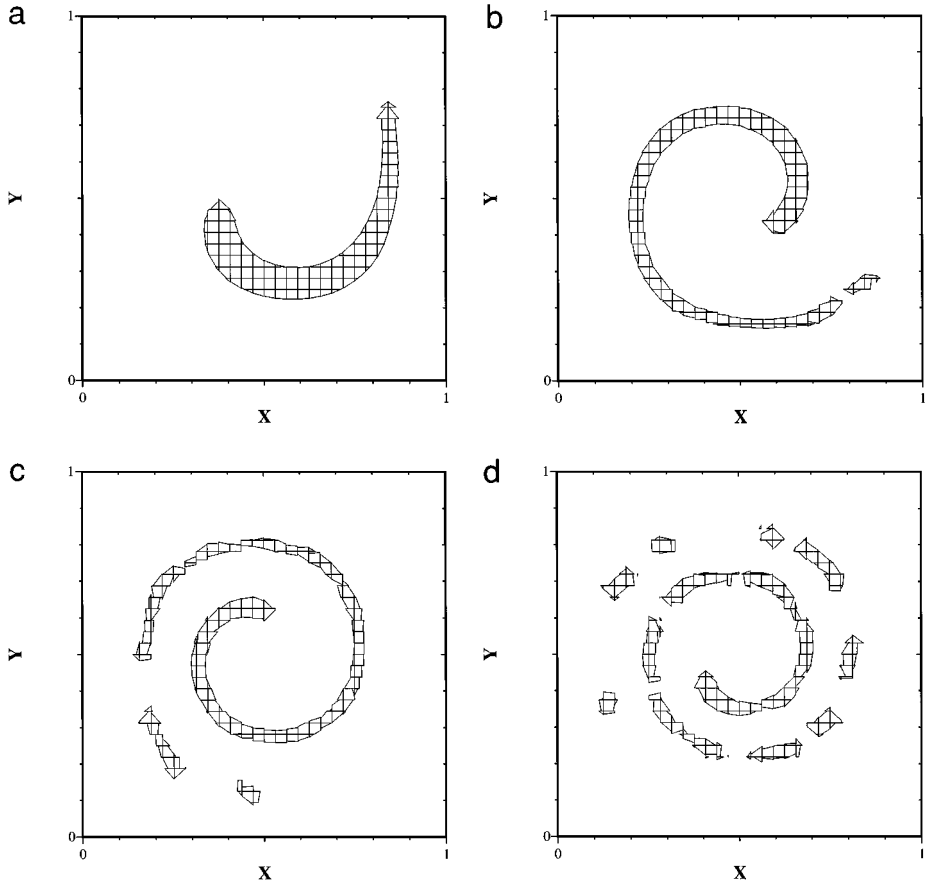

FIG. 15. Results for long time integration of a circular fluid body placed in the single-vortex flow field on a 32^2 grid. (a) $t = 0.75$. (b) $t = 1.50$. (c) $t = 2.25$. (d) $t = 3.00$. (e) $t = 3.75$. (f) $t = 4.50$. (g) $t = 5.25$. (h) $t = 6.00$.

these regions from inaccurate interface normal estimations. Numerical surface tension in these high curvature and thin filament regions can be easily reduced (well below physical levels) with increased refinement.

Deformation field. The deformation velocity field given by (14) poses a test more difficult than the single vortex, forcing the circular fluid body through quite extreme topological changes. Time reversal is again used to obtain quantitative results. The initial position of the circular body falls directly between two vortices; hence, we find that the results fail to converge for long T. This is evident in Table 3, where second-order convergence is realized only for $T = 1.0$. For longer periods, the results converge to only first order. The lack of qualitative similarity between the solutions obtained on the two finest grids at $T = 4$ (Fig. 16) is evidence for the lower convergence.

In Fig. 17 the deformation velocity field shows its ability to tear apart the circular body. Despite the severity of the interface deformation and topology change, mass conservation is maintained and the solution bears a qualitative resemblance to the true solution. At this coarse resolution (32^2), the solution is not high quality, but its qualitative correctness exhibits the robustness we are seeking in an interface tracking method.

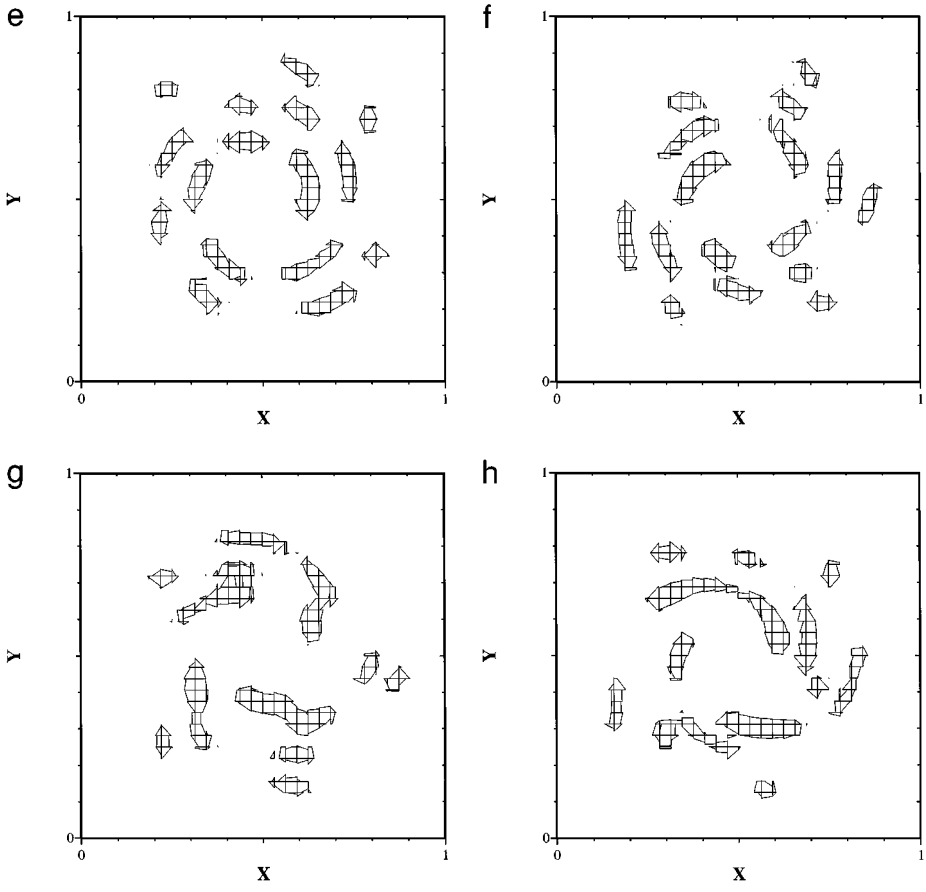

FIG. 15—*Continued*

We conclude our deformation field examples with a brief disgression into operator-split, time-integration methods. All results presented to this point are obtained with our unsplit time-integration scheme detailed in Section 4.4. When an operator-split, time-integration scheme is used for these tests, the observed convergence rates and error norms are similar to the presented unsplit results.

Operator-split, time-integration methods, however, are inferior for two important reasons: efficiency and symmetry preservation. Because volume tracking methods are dominated by

TABLE 3
L_1 Error Norms and Convergence Rates for a Circular Fluid Body Placed in the Time-Reversed, Deformation Flow Field

Grid	Error ($T = 1.0$)	Order ($T = 1.0$)	Error ($T = 2.0$)	Order ($T = 2.0$)	Error ($T = 4.0$)	Order ($T = 4.0$)
32^2	5.20×10^{-3}		1.96×10^{-2}		4.68×10^{-2}	
		1.62		0.81		1.52
64^2	1.69×10^{-3}		1.12×10^{-2}		1.63×10^{-2}	
		1.95		0.91		0.84
128^2	4.36×10^{-4}		5.95×10^{-3}		9.08×10^{-2}	

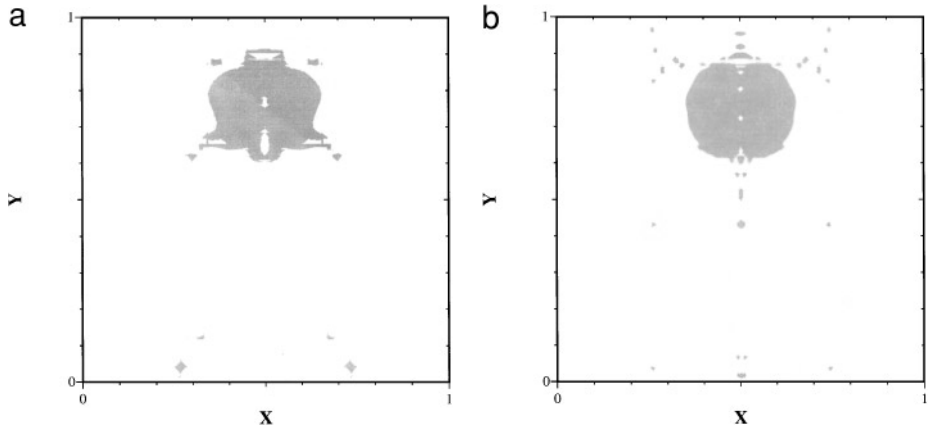

FIG. 16. Results for a circular fluid body placed in the time-reversed, deformation flow field at $t = T$ for $T = 4$. (a) 64^2 grid. (b) 128^2 grid.

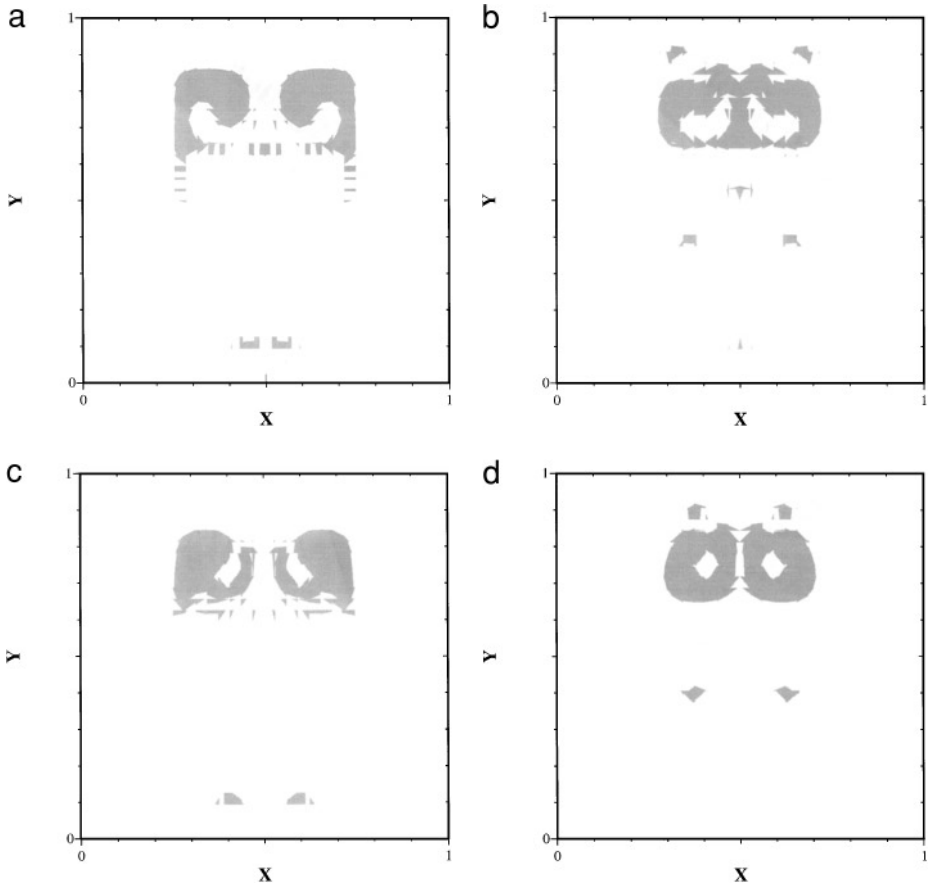

FIG. 17. Results for a circular fluid body placed in the deformation velocity field on a 32^2 grid. (a) $t = 0.50$. (b) $t = 1.00$. (c) $t = 1.50$. (d) $t = 2.00$.

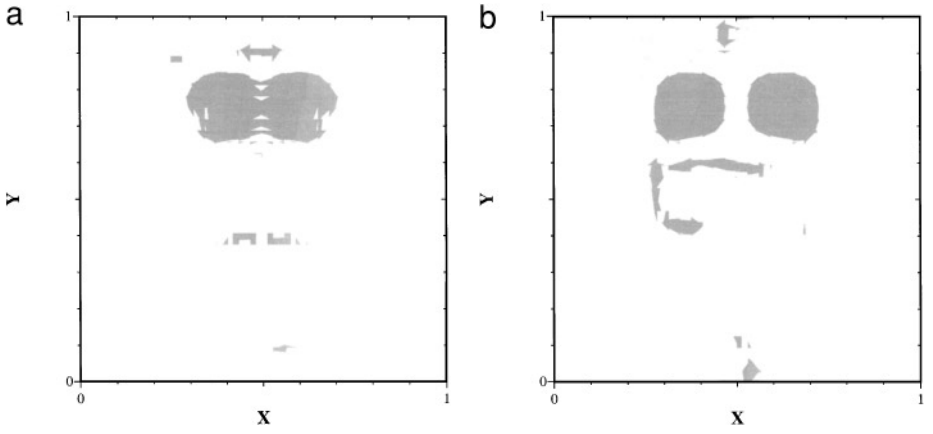

FIG. 18. Results at $t = 4.0$ for a circular fluid body placed in the deformation flow field using an operator-split time-integration method on a 32^2 grid. In (a) the flow is time-reversed with $T = 4$. These results should be compared with the unsplit results in Fig. 19. (a) Time-reversed flow. (b) Non-time-reversed flow.

the cost of reconstructing the interface, an operator-split method is roughly twice as expensive as an unsplit method (in 2D) because one extra reconstruction is required. Operator-splitting also fails to maintain symmetry, even when sweep directions are alternated. This is evident in the results of Fig. 18, which illustrate that operator-split time-integration solutions are inferior to those of our unsplit scheme (shown in Fig. 19). For these reasons we prefer methods based on unsplit time-integration schemes.

The results presented in this section are evidence for the topological changes our volume tracking method must manage while tracking an initially circular fluid body placed in the vortex and deformation flow fields. For performance of other tracking methods on these same problems, see the results in [15]. The methods tested in [15] are those based on particles, level sets [42], and high resolution upwind schemes such as PPM [43].

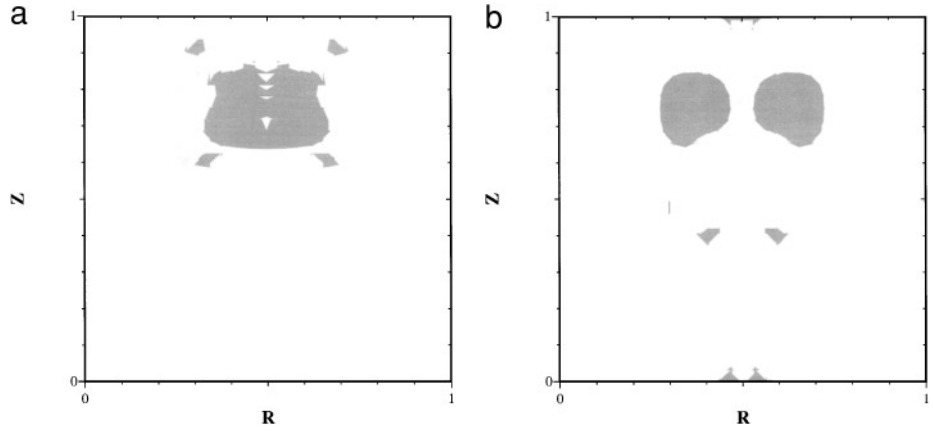

FIG. 19. Results at $t = 4.0$ for a circular fluid body placed in the deformation flow field using an unsplit time-integration method on a 32^2 grid. In (a) the flow is time-reversed with $T = 4$. These results should be compared with the operator split results in Fig. 18. (a) Time-reversed flow. (b) Non-time-reversed flow.

6. CONCLUSIONS

A new second-order-accurate method for the volume tracking of material interfaces in two dimensions has been presented. The algorithm is based upon solutions of material volume evolution equations in which material volume fluxes are estimated geometrically. The volume fluxes are defined as the truncation volumes formed when reconstructed piecewise linear interfaces pass through total volume fluxes bounded by characteristic flow lines. The piecewise linear reconstruction ensures second-order spatial accuracy and the characteristic volume fluxes ensure second-order temporal accuracy (via a multidimensional unsplit time integration scheme). A template of the simple geometric functions needed to compute the material volumes fluxes is provided in detail sufficient enough for implementation. Motivating the development of this method is our need for high-fidelity models for topologically complex interfacial flows; hence, we have integrated our method with flows inducing gross interface topology changes, whereby an initially simple interface configuration is subjected to a variety of controlled vortical and shearing flows. Numerical results for these complex topology tests provide convincing evidence for the algorithm's excellent solution quality and fidelity.

Several aspects of this volume-tracking methodology deserve further attention. Numerical surface tension is brought about when interfaces are approximated as piecewise linear, and this effect needs to be understood and quantified, especially for those interfacial flows where physical surface tension is important. A three-dimensional (piecewise planar) extension of this method that maintains formal second-order accuracy is also of interest, in particular the algorithms for temporal integration and plane normal estimation.

APPENDIX A: FINDING THE INTERFACE NORMAL

Unlike the line constant ρ, the method used determines that the interface normal $\mathbf{n}$ is arbitrary; i.e., it is not constrained by volume conservation considerations. The algorithm used in determining $\mathbf{n}$ is nonetheless crucial, being a key distinguishing aspect of volume-tracking algorithms. Simple estimations for $\mathbf{n}$ can cause a volume tracking algorithm to exhibit overall first-order tendencies, i.e., an inability to reproduce linear interfaces. For example, a method for $\mathbf{n}$ that forces $\mathbf{n}$ to align with mesh logical coordinates yields a low-order volume-tracking algorithm reminiscent of the SLIC method. Other than the temporal integration method (discussed in Sections 4.3. and 4.4), the method used for estimation of $\mathbf{n}$ plays a principal role in the overall accuracy of a volume tracking algorithm. In the following, we will first discuss three methods for computing $\mathbf{n}$, then illustrate below the impact of using $\mathbf{n}$ resulting from these methods on the reconstruction of simple circular and square volume distributions.

A.1. *Youngs' (Least Squares)*

Our first method is an extension of Youngs' second method [21] to general grids [32]. Youngs' second method for estimation of $\mathbf{n}$ was originally developed for 3D tensor-product meshes in which cells are orthogonal bricks having generally different widths. The basic algorithm invokes straightforward (yet wide stencil) finite-difference approximations for the volume fraction gradient (∇f) to define the interface normal $\mathbf{n}$ [21]. It is robust and simple, but unfortunately first-order accurate, capable of reproducing a linear interface only in certain isolated cases.

We have extended Youngs' finite difference approximations for ∇f to 2D or 3D arbitrary-connectivity (fully unstructured) meshes. The method is based on the work of Barth [44], who has devised innovative least squares algorithms for the linear and quadratic reconstruction of discrete data on unstructured meshes. Second (and higher) order accuracy has been demonstrated on highly irregular (e.g., random triangular) meshes. In this approach, volume fraction Taylor series expansions f_i^{TS} are formed from each reference cell volume fraction f_i at point $\mathbf{x}_i$ to each cell neighbor f_k at point $\mathbf{x}_k$. The sum $(f_i^{\mathrm{TS}} - f_k)^2$ over all n immediate neighbors is then minimized in the least squares sense using the normal equations. This L_2 norm minimization will yield the volume fraction gradient ∇f_i as solutions to the linear system (in 2D)

$$A^T A\mathbf{x} = A^T\mathbf{b}, \tag{15}$$

where

$$A = \begin{pmatrix} \omega_k(x_k - x_i) & \omega_k(y_k - y_i) \\ \vdots & \vdots \\ \omega_n(x_n - x_i) & \omega_k(y_n - y_i) \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} \omega_k(f_k - f_i) \\ \vdots \\ \omega_n(f_n - f_i) \end{pmatrix},$$

where $\omega_k = 1/|\mathbf{x}_k - \mathbf{x}_i|^t$ (we take $t = 2$) and

$$\mathbf{x} = \begin{pmatrix} \nabla_x f_i \\ \nabla_y f_i \end{pmatrix}.$$

The normal is then computed as

$$\mathbf{n} = \begin{pmatrix} \dfrac{\nabla_x f_i}{|\nabla f_i|} \\ \dfrac{\nabla_y f_i}{|\nabla f_i|} \end{pmatrix}, \tag{16}$$

where

$$|\nabla f_i| = \sqrt{(\nabla_x f_i)^2 + (\nabla_y f_i)^2}.$$

Here we choose the eight nearest neighbors on a two-dimensional Cartesian grid (that is, all neighbors sharing a vertex).

Least squares reconstruction methods, such as that implied by the linear system above, are quite powerful and attractive. They are not married to any particular mesh topology or dimensionality; hence they are easily amenable to any (unstructured) 2D or 3D mesh. All that is required is a set of discrete data points described by their data values and their physical location.

Despite the attractive characteristics of least squares methods, we have yet to construct a linear system whose solution yields interface normals $\mathbf{n}$ that are generally second-order accurate. We will continue to pursue this methodology, however, because it offers perhaps the best possibility (over other methods) for a generally second-order accurate $\mathbf{n}$ estimation in 3D. Careful construction of data-dependent weights might provide a linear system that yields such a second-order discretization.

A.2. *Error Minimization*

In the second method **n** does not result from an explicit expression, but rather from a least squared error minimization procedure that is iterative (but local). This method is shown to be second-order accurate. The second method for estimation of **n** is based upon the recent volume tracking work of Pilliod [10], in which particular scrutiny is paid to the interface reconstruction step. They conclude that methods for estimating **n** based on a minimization of error (by some measure) are optimal. Developed as a result of this analysis was an algorithm for **n** that, like our first method, is also guided by a least squares prescription for error. In the least squares procedure, reconstructed interface line segments are extrapolated into nearest-neighbor cells (all cells bordering the reference mixed cell), and the nearest-neighbor truncation volumes are computed (in addition to the reference mixed cell). Differences between nearest-neighbor truncation volumes and their actual material volumes are minimized in the least squares sense by iteratively changing the reference mixed cell value of **n** until convergence is achieved. This method is second-order in the sense that it is able to reconstruct linear interfaces exactly.

Unfortunately, the least squares method can be prohibitively expensive, especially in 3D, because interfaces must be reconstructed in all mixed cells bordering the reference mixed cell each iteration that is required to find **n**. For 2D tensor-product Cartesian meshes, however, this expensive iterative procedure can be circumvented with an innovative "fast least squares" (FLS) procedure [10], making the method viable.

A.3. *Swartz's Method*

The final method we consider for estimating **n** is based on the work of Blair Swartz [45]. This iterative procedure, which also preserves linear interfaces (is second-order), was shown recently to converge quadratically on generally irregular 2D meshes [31]. The basic algorithm consists of several steps, shown schematically in Fig. 20. First, a reasonable estimate for **n** is made (e.g., using Youngs' second method); then the interface line segment is reconstructed according to the Interface Reconstruction Template given in Section 4.2 of this paper. Second, this normal is assigned to a selected mixed cell bordering the reference mixed cell, and a linear interface is reconstructed in the border mixed cell (Fig. 20a). Third, a line is defined whose endpoints are the interface centroid in the reference mixed cell and the interface centroid in the border mixed cell (Fig. 20b). The normal to this line (connecting the centroids) defines the new guess for **n**. This process is repeated until convergence.

In applying the Swartz algorithm to interface reconstruction, we have found the following modifications useful. First, the stencil should be symmetric; i.e., all mixed cells bordering the reference mixed cell (in a 3×3 block) should contribute to the new guess for **n**, not just one selected border cell. An inverse-distance weighted contribution to the new guess for **n** is used for all border mixed cells. Our modified Swartz algorithm for **n** proceeds according to the following template:

Modified Swartz Interface Normal Template

1. Guess a normal **n** for the interface line in the reference mixed cell.
2. Given **n**, compute the interface line position in the reference mixed cell by finding the interface constant ρ in (4).
3. Using the reference mixed cell normal **n**, compute interface line positions in all mixed cells bordering the reference mixed cell (within a 3×3 stencil).

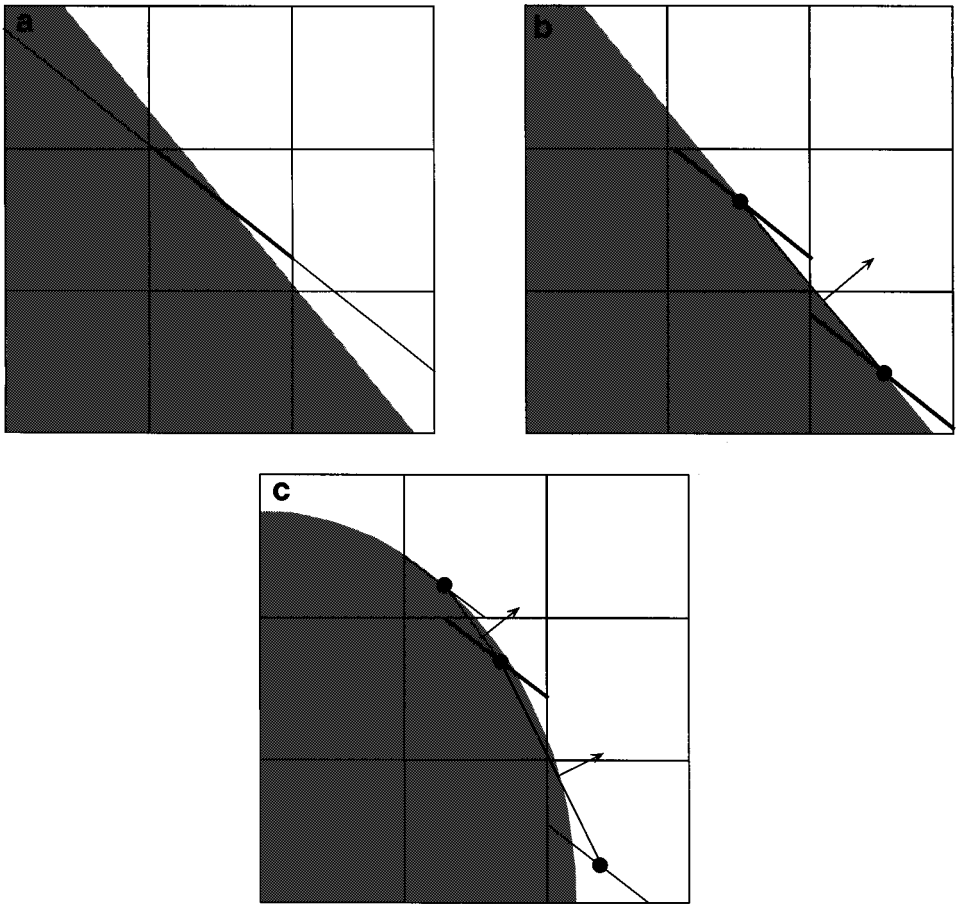

FIG. 20. Illustration of the basic Swartz algorithm steps required to find the interface normal **n** in a reference mixed cell, shown here as the center cell of a 3×3 stencil. (a) An initial guess for **n** results in an initial reconstructed interface line in the reference mixed cell. (b) A new guess for **n** is the normal to the line connecting interface line centroids of the reference cell and a selected bordering mixed cell. (c) If the interface has curvature, multiple bordering mixed cells are needed to find the new guess for **n** 31.

4. Connect the interface line centroid in the neighboring mixed cell with an interface line centroid in the cell being reconstructed. Repeat for all mixed cells in the stencil.
5. Define a new normal $\mathbf{n}_{\text{new}}$ as the average of all of the lines formed in the previous step (using an appropriate weighting such as the inverse distance of the neighboring line centroid to the cell being reconstructed).
6. If $\mathbf{n}_{\text{new}}$ differs from **n** by some prescribed tolerance, go to step (2).

End template

A modified Swartz algorithm based on the template above yields a second-order normal **n** for generally nonlinear interfaces (those having curvature). This algorithm performs equally well in 2D Cartesian and azimuthally symmetric cylindrical coordinate systems. It can also find a second-order normal on meshes more general than the tensor-product meshes required for Youngs' second method or the least squares minimization method. For example, it has been applied successively on 2D nonorthogonal, unstructured meshes [31].

Extension to three dimensions also appears to be a straightforward exercise. The modified Swartz method does not, however, lack restrictions or disadvantages. First, it can be prohibitively expensive, especially in 3D, because interfaces must be reconstructed in all mixed cells bordering the reference mixed cell, for each iteration required to find **n** (just like the least squares method). Second, it is outperformed by Youngs' second method and the least squares method when the interface is poorly resolved (because of the lack of bordering mixed cells). Otherwise, it is close in behavior to the procedure described next, and, as such, will not be explored in detail for dynamic problems.

A.4. *Reconstruction Accuracy*

The accuracy of these interface normal algorithms is most easily assessed by analyzing estimates of **n** operating on discrete volume data that replicates known (exact) interface geometries, such as a circle or square. If the various **n** estimates are used to perform piecewise linear interface reconstructions on the data, the differences and errors can be visualized, as shown in the next section.

We now assess the accuracy of the interface normal algorithms detailed in the previous section by applying them to known circular and square distributions of volume data. To avoid artificial regularity of the results, we offset the circle and square centers (relative to mesh logical coordinates). For all tests, we partition a unit square domain with from 10^2 to 160^2 orthogonal, uniform cells. Discrete volume data is initialized on the domain to replicate either a circle (radius 0.368) centered at (0.525, 0.464) or a square (side length 0.512) centered at (0.468, 0.541). The discrete volume initialization is exact in every cell, so we expect computational values of the interface normals **n** to converge to the analytical values as the mesh is refined.

Graphical pictures of the piecewise linear interface reconstructions enable a qualitative assessment of each **n** algorithm, as shown in Figs. 21–23. Results are displayed for the coarsest grid, where "reconstruction errors" are most evident. Quantitative error assessments are possible for these known geometries if one defines the reconstruction error as the volume integral of the difference between actual interface and the linear representation. With errors quantified, convergence rates can be computed, as shown in Tables 4–6. For the circular

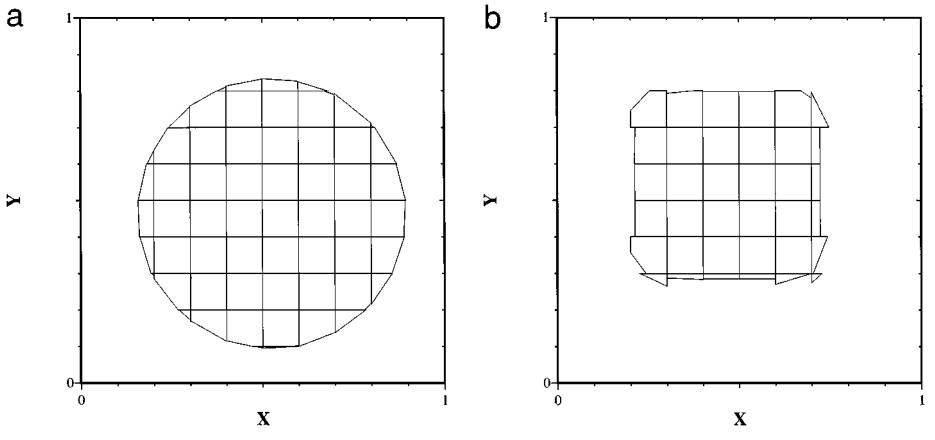

FIG. 21. Piecewise linear reconstructed interfaces based on Youngs' second method for the interface normal **n** on a 10×10 grid. (a) Offset circle. (b) Offset square.

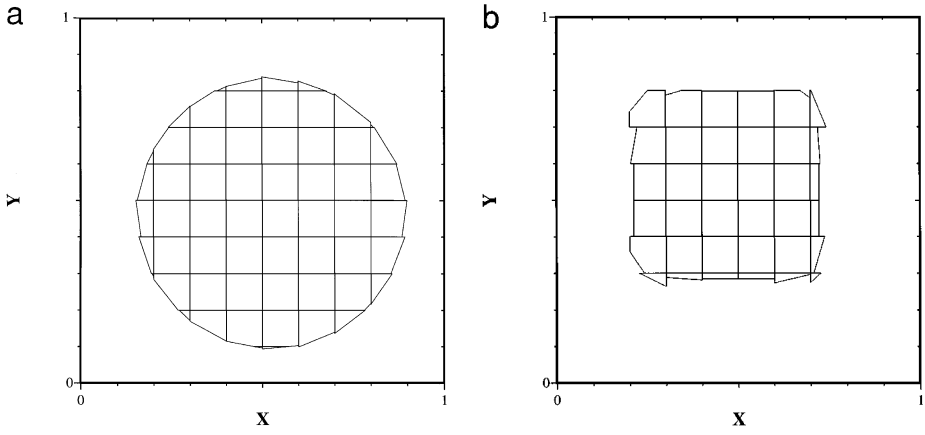

FIG. 22. Piecewise linear reconstructed interfaces based on Swartz's method for the interface normal **n** on a 10×10 grid. (a) Offset circle. (b) Offset square.

distribution, which has no curvature singularities, we expect each interface normal method to converge. For the square, on the other hand, curvature singularities (the corners) will preclude convergence. Along with inhibiting convergence, curvature singularities will also be the site of the most flagrant reconstruction errors.

The errors measured on these simple tests give rise to some interesting observations. Youngs' second method (which is first order) surprisingly exhibited the lowest errors on the coarsest grids, but its lower convergence rate eventually led to errors that exceeded the other methods as the grid is refined. In general, for large curvatures (relative to mesh spacing), Youngs' second method is quite robust, which is in contrast to the second-order Swartz method, as discussed later.

Results for Youngs' second method are given in Table 4 and Fig. 21. The convergence rate deteriorates as the grid is refined, approaching first-order. A qualitative assessment of the reconstruction (Fig. 21) is favorable, but the square corners present the method with more problems than the second-order Swartz or least squares methods.

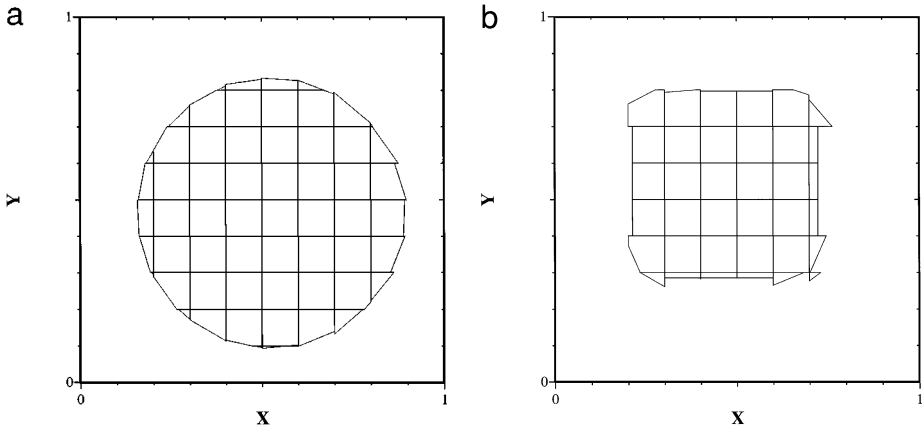

FIG. 23. Piecewise linear reconstructed interfaces based on the least squares method for the interface normal **n** on a 10×10 grid. (a) Offset circle. (b) Offset square.

TABLE 4
L_1 Circle Reconstruction Error Norms Using Youngs' Second Method for the Interface Normal n

Grid	Error	Order
10^2	2.04×10^{-3}	
		1.59
20^2	6.79×10^{-4}	
		1.38
40^2	2.60×10^{-4}	
		1.36
80^2	1.01×10^{-4}	
		0.93
160^2	5.29×10^{-5}	

Swartz's method produces second-order results, as is evident in Table 5. Absolute errors, however, are larger than those exhibited by Youngs' second method on the coarser grids. This can be seen in the quality of the coarse-grid reconstruction shown in Fig. 22. The piecewise linear segments along the circle (Fig. 22a) are not as continuous as those generated with Youngs' second method in Fig. 21. This coarse-grid degradation may be indicative of the problems encountered with Swartz's method on difficult transport problems, where fluid bodies with complex topologies (high curvatures) can be created.

Finally, consider the reconstruction errors resulting from the least squares error method. This method is also second-order, as seen in Table 6. While the absolute errors in Table 6 are slightly larger than the Swartz method, the reconstruction solutions (Fig. 23) are judged to be superior for both the circle and the square. The reconstruction is more continuous for the circle and the singularities in the corners are more localized.

The cost of each reconstruction technique varies with the second-order accurate methods being much more expensive than the first-order methods. Youngs' method is approximately five times cheaper than the fast least squares minimization. In turn, the fast least squares is slightly less than three times as cheap as the full least squares minimization or Swartz's method.

TABLE 5
L_1 Circle Reconstruction Error Norms Using Swartz's Method for the Interface Normal n

Grid	Error	Order
10^2	2.71×10^{-3}	
		1.98
20^2	6.84×10^{-4}	
		2.40
40^2	1.29×10^{-4}	
		2.07
80^2	3.08×10^{-5}	
		2.22
160^2	6.63×10^{-6}	

TABLE 6
L_1 Circle Reconstruction Error Norms Using the Least Squares Method for the Interface Normal n

Grid	Error	Order
10^2	3.21×10^{-3}	
		2.18
20^2	7.08×10^{-4}	
		2.29
40^2	1.45×10^{-4}	
		1.94
80^2	3.78×10^{-5}	
		2.01
160^2	9.38×10^{-6}	

APPENDIX B: TRANSLATION AND ROTATION TESTS

Since translational and rotational flows do not induce topology change, the volume fractions associated with fluid bodies entrained in these flows are known exactly. In such cases, error norms can be defined based on some positive-definite measure of the differences observed between the computed and exact values of f. We choose to estimate computed errors in these problems with an L_1 norm defined as

$$E^{L_1} = \sum_{grid} V_{i,j} \left| f_{i,j}^{\text{computed}} - f_{i,j}^{\text{exact}} \right|. \tag{17}$$

The E^{L_1} error defined above has units of area (or volume in 3D); therefore, its change with mesh size can be used to infer rates of convergence.

B.1. *Test Problems*

For both the translation and rotation test problems, a circular fluid body is placed in a unit square computational domain that is partitioned with either 32^2, 64^2, or 128^2 orthogonal, uniform cells. The circular body is represented with a scalar field that is unity and zero inside and outside the circle, respectively. For those cells containing the circular interface, the scalar field is set to a value between zero and one, in proportion to the cell volume truncated by the circle. This field represents a characteristic (or color) function, which for our purposes is the fluid volume fraction for a circular fluid body, i.e., the volume fraction is 100% inside the circle and 0% outside. All boundaries are periodic. For these tests, we use the unsplit time-integration scheme given by (11) and the least squares method for estimating the interface normal **n**.

Simple translation. A uniform and constant velocity field having positive, equal components is imposed everywhere in the domain. This solenoidal field will cause fluid bodies to translate diagonally across the mesh at a 45° angle. The circular fluid body (radius 0.25), initially centered at (0.50, 0.50), should return to its initial position after 1 time unit, allowing error measurement with (17). A CFL number of $\frac{1}{2}$ is used. The body should not change shape as a result of this movement.

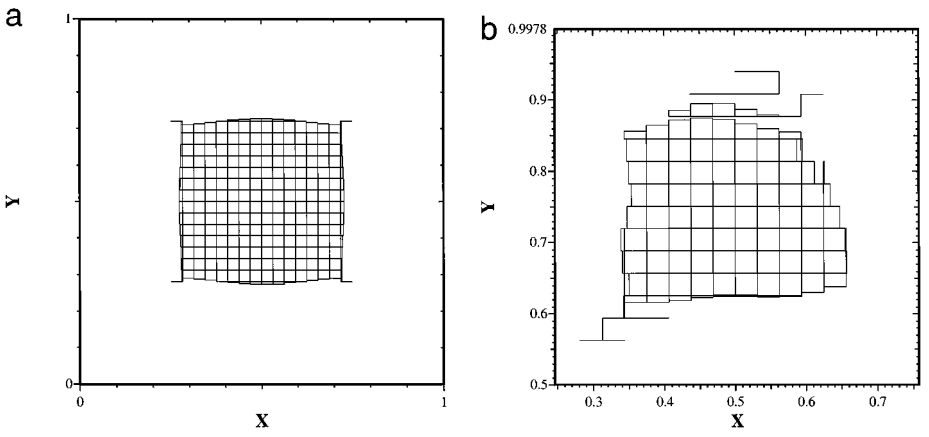

FIG. 24. Performance of a piecewise constant tracking method in translating and rotating an initially circular fluid body on a 32^2 grid. Reconstructed interfaces are shown for the body (a) after being translated at 45° for a distance of two domain diagonals, and (b) after being rotated one revolution. (a) Translation. (b) Rotation (magnified view).

Solid body rotation. A constant-vorticity velocity field is imposed at the center of the domain. This solenoidal field will cause all fluid bodies to rotate around this center. The circular fluid body (radius 0.15), initially centered at (0.50, 0.75), should return to its initial position after π time units, allowing error measurement with (17). A CFL number of $\frac{1}{2}$ is used, based on the maximum velocity in the domain. The body should not change shape as a result of this rotation.

B.2. *Results*

Let us first assess the ability of traditional piecewise constant volume tracking methods to translate and rotate the circular body. We employ a piecewise constant method that represents a combination of the SLIC and VOF methods.[3] The circle is shown after translating two diagonals of the computational domain in Fig. 24a and after rotating one revolution in Fig. 24b. In both cases the results are inferior to the piecewise linear results depicted in Figs. 25 and 26. Computed piecewise constant L_1 errors are one to two orders of magnitude larger than the coarsest-grid (32^2) piecewise linear results, with the differences becoming even larger as the grid is refined. Convergence for this piecewise constant scheme is at best first order.

Two important features are evident in Fig. 24 that are typical of piecewise constant volume tracking methods. First, the circular body in Fig. 24a remains in one piece after translation, but the circle's topology has become square-like by virtue of its interfaces aligning with the grid coordinates. This misalignment is the direct result of the imposed horizontal and vertical interface reconstructions associated with piecewise constant approximations. Second, after rotating the circular body one revolution in Fig. 24b, the circle has lost small bits of fluid that have been unphysically ejected from the main body. This unacceptable flotsam creation always accompanies a piecewise constant method when the flowfield has appreciable spatial variation.

[3] Like the VOF method, an interface normal **n** computed from a 3×3 stencil is used to determine interface orientation. Like the SLIC method, the interface is then reconstructed vertically or horizontally, depending upon the relative magnitudes of the **n** components.

TABLE 7

L_1 Error Norms and Convergence Rates for a Circular Fluid Body Translated Two Domain Diagonals at Three Different Angles to the Grid

Grid	Error (0°)	Order (0°)	Error (26.565°)	Order (26.565°)	Error (45°)	Order (45°)
32^2	1.97×10^{-4}		1.99×10^{-3}		6.21×10^{-4}	
		2.67		2.16		1.33
64^2	3.09×10^{-5}		4.45×10^{-4}		2.47×10^{-4}	
		2.50		2.03		2.27
128^2	5.48×10^{-4}		1.09×10^{-4}		5.10×10^{-5}	

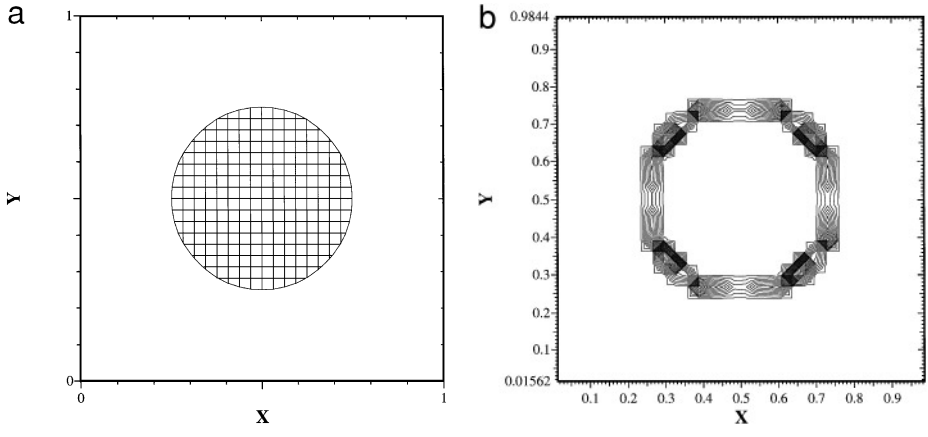

FIG. 25. Performance of the piecewise linear volume tracking method in translating an initially circular fluid body at 45° for a distance of two domain diagonals on a 32^2 grid. (a) Reconstructed interfaces. (b) L_1 error contours.

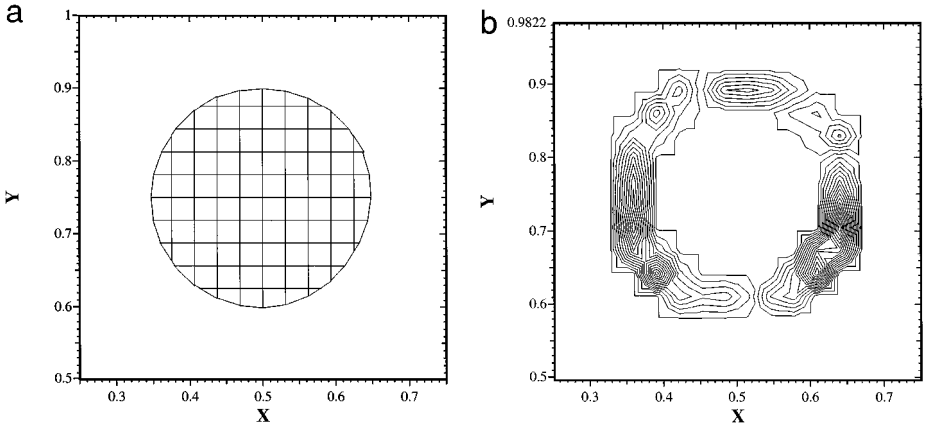

FIG. 26. Performance of the piecewise linear volume tracking method in rotating an initially circular fluid body for one revolution on a 32^2 grid. (a) Reconstructed interfaces. (b) L_1 error contours.

TABLE 8
L_1 Error Norms and Convergence Rates for a Circular Fluid Body Rotated One Revolution

Grid	Error	Order
32^2	1.61×10^{-3}	
		2.19
64^2	3.54×10^{-4}	
		1.98
128^2	8.95×10^{-5}	

Recall that our second-order unsplit time integration is used for the piecewise constant method that generated the results in Fig. 24, so accuracy degradation is solely due to the piecewise constant geometry approximation. This crude approximation fails to pass simple translation tests, as shown in Fig. 24.

Our piecewise linear scheme, on the order hand, preserves the circular shape after translation, as shown in Fig. 25. The reconstructed interfaces and error contours are isotropic with respect to flow, i.e. exhibiting no bias toward flow direction. Errors are slightly higher at 45° angular increments around the circle (Fig. 25b). Mass conservation is exact. Convergence rates based on (17) are generally second order, but they exhibit some dependence on the flow direction (relative to the grid), as shown in Table 7. The convergence rate is clearly second order in most cases, with the highest errors generated by the 26.565° translation. These excellent translation results should be expected from a useful interface tracking method. Excellent translation performance is necessary, *but not sufficient*, for a method required to track interfaces in complex topology flows.

Solid body rotation results add little additional insight relative to the translation results, except that phase errors are now more apparent, as shown in Fig. 26b. Second order accuracy is again exhibited, as shown in Table 8. Neither the translation or rotation problems pose serious problems for a well-designed interface tracking method. They will expose diffusion and dispersion problems typical of standard advection methods, but are not sufficient for methods designed specifically for interfaces. For both translation and rotation, the circular shape is preserved; hence we can declare victory *for now*, but more stringent tests are necessary.

ACKNOWLEDGMENTS

We thank Jay Mosso, Gerry Puckett, Greg Miller, Blair Swartz, Len Margolin, and our referees for their numerous helpful comments and suggestions in completing this work.

REFERENCES

1. G. Strang, On the construction and comparison of difference schemes, *SIAM J. Numer. Anal.* **5**, 506 (1968).
2. S. T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, *J. Comput. Phys.* **31**, 335 (1979).
3. R. DeBar, *Fundamentals of the KRAKEN Code*, Technical Report UCIR-760, LLNL (1974).
4. W. F. Noh and P. R. Woodward, SLIC (simple line interface method), in *Lecture Notes in Phys., Vol. 59*, edited by A. I. van de Vooren and P. J. Zandbergen (Springer-Verlag, Berlin/New York, 1976), p. 330.

5. C. W. Hirt and B. D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, *J. Comput. Phys.* **39**, 201 (1981).
6. A. J. Chorin, Flame advection and propagation algorithms, *J. Comput. Phys.* **35**, 1 (1980).
7. P. K. Barr and W. T. Ashurst, *An Interface Scheme for Turbulent Flame Propagation*, Technical Report SAND82-8773, Sandia National Laboratories (1984).
8. N. Ashgriz and J. Y. Poo, FLAIR—flux line-segment model for advection and interface reconstruction, *J. Comput. Phys.* **93**, 449 (1991).
9. D. L. Youngs, Time-dependent multi-material flow with large fluid distortion, in *Numerical Methods for Fluid Dynamics*, edited by K. W. Morton and M. J. Baines (Academic Press, New York, 1982), p. 273.
10. J. E. Pilliod Jr., *An Analysis of Piecewise Linear Interface Reconstruction Algorithms for Volume-of-Fluid Methods*, Master's thesis, University of California at Davis, 1992.
11. J. E. Pilloid Jr. and E. G. Puckett, Second-order volume-of-fluid algorithms for tracking material interfaces, *J. Comput. Phys.*, submitted.
12. B. D. Nichols and C. W. Hirt, *Methods for Calculating Multi-Dimensional, Transient Free Surface Flows Past Bodies*, Technical Report LA-UR-75-1932, Los Alamos National Laboratory (1975). [Appeared in the First International Conference on Numerical Ship Hydrodynamics, Gaithersburg, MD, 10/75]
13. C. W. Hirt and B. D. Nichols, A computational method for free surface hydrodynamics, *Journal of Pressure Vessel Technology* **103**, 136 (1981).
14. M. L. Norman and K.-H. A Winkler, 2-D Eulerian hydrodynamics with fluid interfaces, in *Astrophysical Radiation Hydrodynamics*, edited by K.-H. A Winkler and M. L. Norman, 1986, p. 187.
15. W. J. Rider and D. B. Kothe, Stretching and tearing interface tracking methods, in *The 12th AIAA CFD Conference, San Diego, June 20, 1995*, AIAA-95-1717.
16. R. L. Bowers and J. R. Wilson, *Numerical Modeling in Applied Physics and Astrophysics* (Jones & Bartlett, Boston, 1991).
17. R. L. Bowers, personal communication.
18. M. Rudman, Volume tracking methods for interfacial flow calculations, *International Journal for Numerical Methods in Fluids* **24**, 671 (1997).
19. J. P. Boris and D. L. Book, Flux-corrected transport I. SHASTA, a fluid transport algorithm that works, *J. Comput. Phys.* **11**, 38 (1973).
20. B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, and G. Zanetti, Modelling merging and fragmentation in multiphase flows with SURFER, *J. Comput. Phys.* **113**, 134 (1994).
21. D. L. Youngs, *An Interface Tracking Method for a 3D Eulerian Hydrodynamics Code*, Technical Report 44/92/35, AWRE (1984).
22. F. L. Addessio, D. E. Carroll, J. K. Dukowicz, F. H. Harlow, J. N. Johnson, B. A. Kashiwa, M. E. Maltrud, and H. M. Ruppel, *CAVEAT: A Computer Code for Fluid Dynamics Problems with Large Distortion and Internal Slip*, Technical Report LA-10613-MS, Los Alamos National Laboratory (1986).
23. K. S. Holian, S. J. Mosso, D. A. Mandell, and R. Henninger, *MESA: A 3-D Computer Code for Armor/Anti-Armor Applications*, Technical Report LA-UR-91-569, Los Alamos National Laboratory (1991).
24. D. B. Kothe, J. R. Baumgardner, S. T. Bennion, J. H. Cerutti, B. J. Daly, K. S. Holian, E. M. Kober, S. J. Mosso, J. W. Painter, R. D. Smith, and M. D. Torrey, *PAGOSA: A Massively-Parallel, Multi-Material Hydro-Dynamics Model for Three-Dimensional High-Speed Flow and High-Rate Deformation*, Technical Report LA-UR-92-4306, Los Alamos National Laboratory (1992).
25. J. S. Perry, K. G. Budge, M. K. W. Wong, and T. G. Trucano, Rhale: A 3-D MMALE code for unstructured grids, in *Advanced Computational Methods for Material Modeling, AMD-Vol. 180/PVP-Vol. 268* (ASME, New York, 1993), p. 159.
26. E. G. Puckett, A volume of fluid interface tracking algorithm with applications to computing shock wave rarefraction, in *Proceedings, 4th International Symposium on Computational Fluid Dynamics, 1991*, p. 933.
27. P. Colella, L. F. Henderson, and E. G. Puckett, A numerical study of shock wave refractions at a gas interface, in *Proceedings of the AIAA Ninth Computational Fluid Dynamics Conference*, edited by T. Pulliam, 1989, p. 426. [AIAA Paper 89-1973-CP]

28. E. G. Puckett and J. S. Saltzman, A 3D adaptive mesh refinement algorithm for multimaterial gas dynamics, *Physica D* **60**, 84 (1992).
29. M. J. Berger and P. Colella, Local adaptive mesh refinement for shock hydrodynamics, *J. Comput. Phys.* **82**, 64 (1989).
30. J. Bell, M. Berger, J. S. Saltzman, and M. Welcome, Three dimensional adaptive mesh refinement for hyperbolic conservation laws, *SIAM Journal on Scientific Computing* **15**, 127 (1994).
31. S. J. Mosso, B. K. Swartz, D. B. Kothe, and R. C. Ferrell, A parallel, volume tracking algorithm for unstructured meshes, in *Parallel Computational Fluid Dynamics '96, Italy, 1996*, edited by P. Schiano.
32. D. B. Kothe, W. J. Rider, S. J. Mosso, J. S. Brock, and J. I. Hochstein, *Volume Tracking of Interfaces having Surface Tension in Two and Three Dimensions*, Technical Report AIAA 96-0859, AIAA (1996). [Presented at the 34rd Aerospace Sciences Meeting and Exhibit]
33. W. J. Rider, D. B. Kothe, S. J. Mosso, J. H. Cerruti, and J. I. Hochstein, *Accurate Solution Algorithms for Incompressible Multiphase Fluid Flows*, Technical Report AIAA 95-0699, AIAA (1995). [Presented at the 33rd Aerospace Sciences Meeting and Exhibit, Reno NV, Jan 9–12 (1995) (also see LANL Report LA-UR-94-3611), also available on World Wide Web at `http://www-xdiv.lanl.gov/XHM/personnel/wjr/Web_papers/pubs.html`]
34. D. B. Kothe, *et al. Computer Simulation of Metal Casting Processes: A New Approach*, Technical Report LALP-95-197, Los Alamos National Laboratory (1995).
35. E. G. Puckett, A. S. Almgren, J. B. Bell, D. L. Marcus, and W. J. Rider, A second-order projection method for tracking fluid interfaces in variable density incompressible flows, *J. Comput. Phys.* **130**, 269 (1997).
36. J. O'Rourke, *Computational Geometry in C*, Cambridge, 1993.
37. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran*, Cambridge, 1986.
38. J. B. Bell, P. Colella, and H. M. Glaz, A second-order projection method of the incompressible Navier–Stokes equations, *J. Comput. Phys.* **85**, 257 (1989).
39. J. K. Dukowicz, *New Methods for Conservative Rezoning (Remapping) for General Quadrilateral Meshes in Rezoning Workshop 1983*, Technical Report LA-10112-C, Los Alamos National Laboratory (1984).
40. P. K. Smolarkiewicz, The multi-dimensional Crowley advection scheme, *Monthly Weather Review* **110**, 1968 (1982).
41. R. J. Leveque, High-resolution conservative algorithms for advection in incompressible flow, *SIAM J. Numer. Anal.* **33**, 627 (1996).
42. M. Sussman, P. Smereka, and S. Osher, A level set approach for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* **114**, 146 (1994).
43. P. Colella and P. Woodward, The piecewise parabolic method (PPM) for gas-dynamical simulations, *J. Comput. Phys.* **54**, 174 (1984).
44. T. J. Barth, *Aspects of Unstructured Grids and Finite-Volume Solvers for Euler and Navier–Stokes Equations* (1995). [VKI/NASA/AGARD Special Course on Unstructured Grid Methods for Advection Dominated Flows AGARD Publication R-787]
45. B. Swartz, The second-order sharpening of blurred smooth borders, *Mathematics of Computation* **52**, 675 (1989).